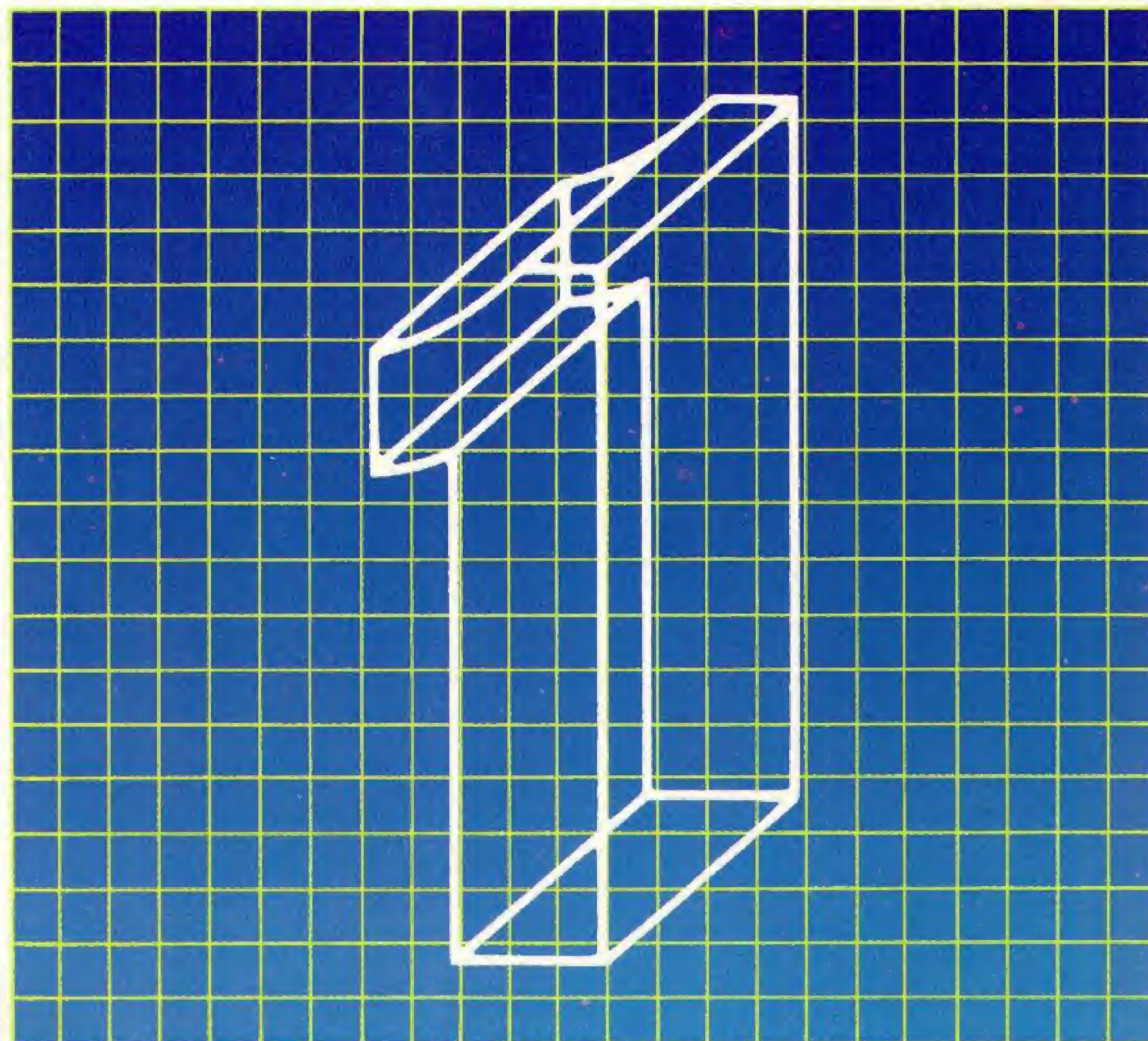
BIBLIOTECA PRACTICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

TALLER DE INFORMATICA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



¿UN TALLER DE INFORMATICA?

ATREVASE A HACER BRICOLAGE CON SU O.P.

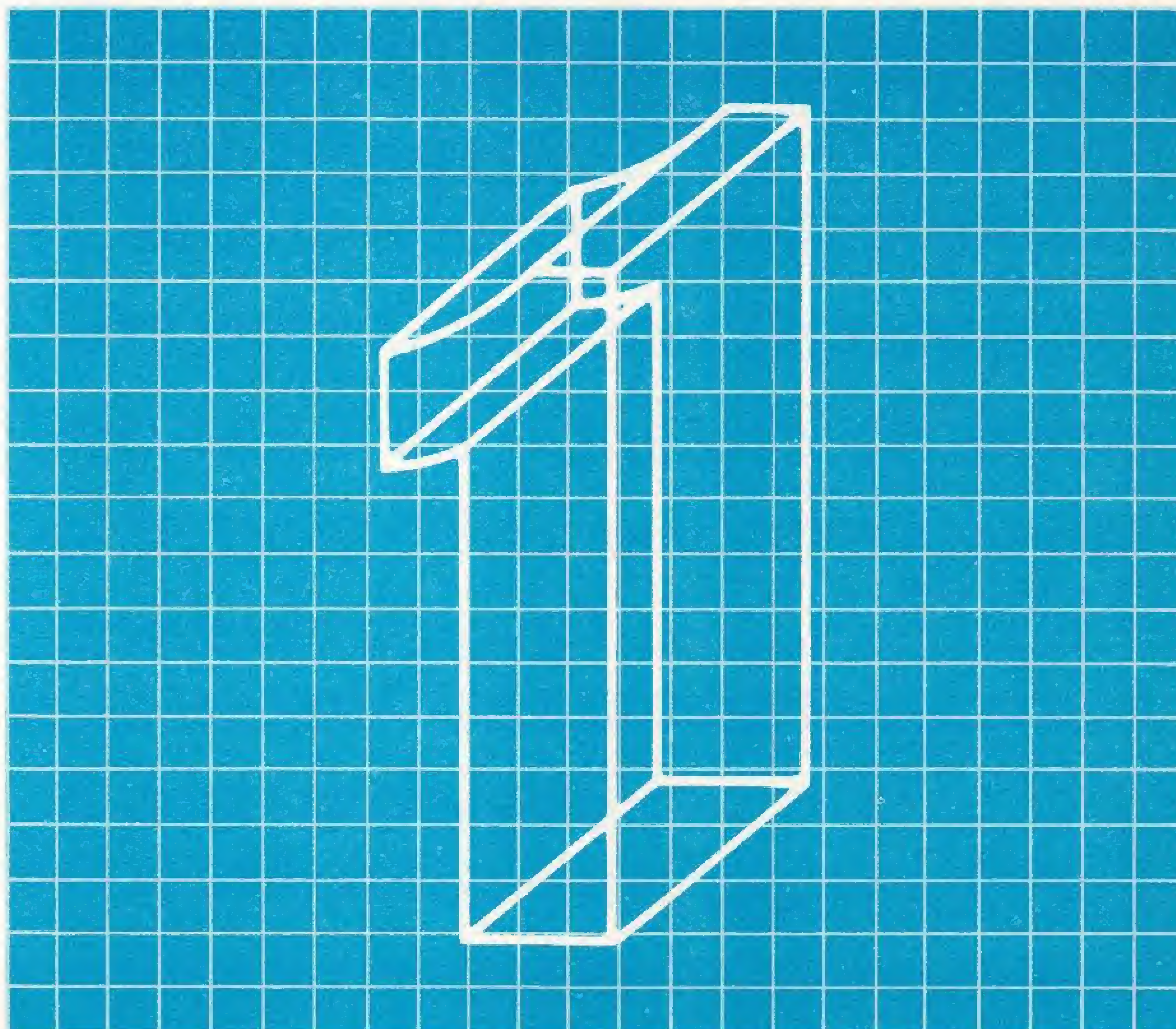
EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Director de la coleccióin:
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:
BRAVO-LOFISH.

Maquetación:
D. SIMON

Dibujos:
JOSE OCHOA

Tomo 1. «Experiencias prácticas en logo», Carlos Albert, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales, Técnico de Informática. «Aprender con el ordenador», AULA DE INFORMATICA APLICADA: Fernando Suero, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Francisco Blanca, Diplomado en Telecomunicación; María José Hernando, Diplomada en Informática; Soledad Tamariz, Diplomada en Telecomunicación. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:
Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-048-0
ISBN de la obra: 84-7688-047-2

Fotocomposición:
ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. PINTO (Madrid).

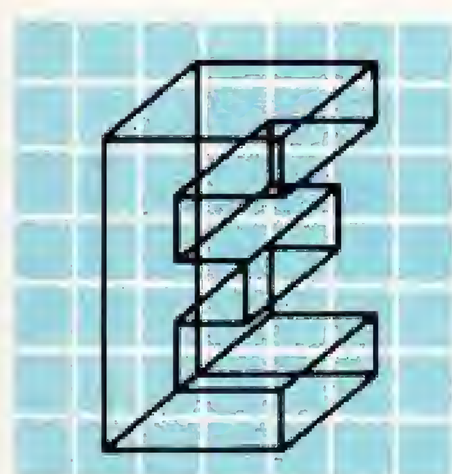
Depósito legal: M-43.185-1986
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S. A.
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid.

Enero, 1987.

INDICE

EXPERIENCIA Y PRACTICAS EN LOGO



L LOGO es un Lenguaje de Programación diferente. Con él podemos dibujar, escribir, hacer cálculos matemáticos y un montón de cosas más, que ya iremos descubriendo.

Es un Lenguaje sencillo y entretenido. Posee un personaje simpático con el que podemos hacer infinidad de cosas. Es la TORTUGA del LOGO.

Esta Tortuga es capaz de pintar, borrar, rellenar, saltar, va a poder orientarse hacia todas las direcciones y se va a poder colocar en cualquier posición de la pantalla.

Te voy a presentar a la Turtuga de Logo.

Coge tu ordenador y carga el Logo. Una vez que se haya cargado te aparecerá la siguiente pantalla:



En ella puedes observar varios mensajes.

Uno de ellos dice la versión de qué se trata y por quién está realizada.

A continuación el mensaje de "Copyrigh" nos dice simplemente que el programa no puede ser copiado de una forma legal.

Luego el Logo nos da la bienvenida y nos indica mediante la aparición de una interrogación y del cursor, que tu ordenador ya está dispuesto a recibir órdenes. No te preocupes si alguna de las órdenes que le das no es correcta; el Logo te lo dirá y te avisará dónde y qué tipo de error has cometido.

En la mayoría de los ordenadores, esta pantalla de presentación es prácticamente igual.

Teclea e introduce la siguiente orden al ordenador:

?MUESTRATORTUGA 

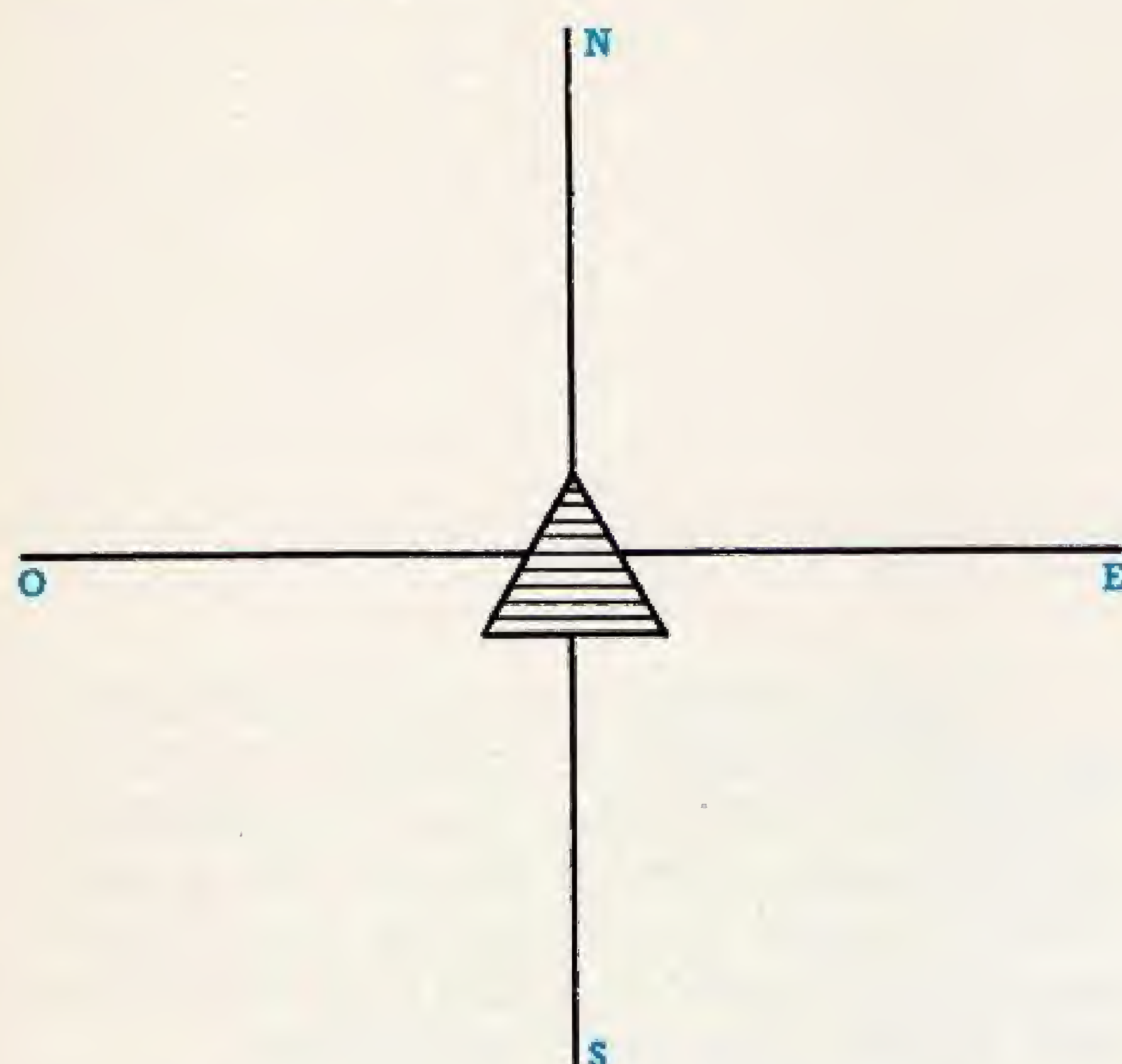
Te aparecerá en el centro de la pantalla la famosa Tortuga.



La Tortuga no es tal, es simplemente un triángulo que simula la forma de una tortuga. Aparece en el centro de la pantalla y orientada hacia arriba. Es importante que te fijes en la punta más pronunciada del triángulo. Va a ser la que señala en todo momento la orientación o rumbo que tiene la Tortuga. Siempre se moverá en la dirección en la que esté orientada esta punta.

Con el Logo puedes escribir tus órdenes en castellano.

Se podrán comparar con la flecha que tienen las brújulas:



El Norte coincidiría con 0° o 360°. El Este con 90° el Sur con 180 y el Oeste con 270°.

Ahora teclea e introduce lo siguiente:

?PANTALLAMIXTA 

Observarás que se ha borrado todo lo que había escrito y el cursor se ha situado más abajo. Tienes sólo unas cuantas líneas abajo para escribir tus órdenes.



Ya estamos dispuestos para ver lo que es capaz de hacer la Turtuga.

Introduce las siguientes órdenes:

NUESTRO MENSAJE DE BIENVENIDA

SELECCION DE PANTALLA
Y ESTADO DE LA TORTUGA

```
? PANTALLAMIXTA
? BP
? MT
```

COLOCACION DE LA TORTUGA
EN LA POSICION DE COMIENZO

```
? GI 90
? SL
? AV 125
? GD 90
? BL
```

DIBUJO LA 1.ª LETRA

```
? AV 50 RE25 GD 90 AV 50
? GI 90 AV 25 RE 50
```

LA POSICIONO EN EL COMIENZO
DE LA 2.ª LETRA

```
? SL
? GD 90
? AV 15
? GI 90
? BL
```

DIBUJO LA 2.ª LETRA

```
? REPITE 4 [AV 50 GD 90]
? GD 90 AV 50 GI 90
```

LA POSICIONO EN EL COMIENZO
DE LA 3.ª LETRA

```
? SL
? GD 90
? AV 15
? GI 90
? BL
```

DIBUJO LA 3.ª LETRA

```
? AV 50 RE 50
? GD 90 AV 50
? GI 90
```

LA POSICIONO EN EL COMIENZO
DE LA 4.ª LETRA

```
? SL
? GD 90
? AV 15
? GI 90
? BL
```

No te olvides de pulsar RETURN después de escribir una orden.

DIBUJO LA 4.ª LETRA

? REPITE E [AV 50 GD 90]
? GI 90 RE 25
? GI 90 RE 50

Si quieres puedes sustituir la tercera orden que has introducido (**MT**) por **OT**.

¿Qué ha ocurrido?

Lo mismo, ¿verdad? Pero, ¿y la tortuga? Ha desaparecido.

Por si todo este lío de órdenes que has introducido han despertado tu interés, analízalas de nuevo sabiendo que:

MT equivale a poner MUESTRATORTUGA.

OT equivale a poner OCULTATORTUGA.

BP equivale a poner BORRAPANTALLA.

GD equivale a poner GIRADERECHA.
GI equivale a poner GIRAIZQUIERDA.
AV equivale a poner AVANZA.
RE equivale a poner RETROCEDE.
BL equivale a poner BAJALAPIZ.
SL equivale a poner SUBELAPIZ.

Aquí está REPITE, ya puedes olvidarte de lo pesado que es hacer una misma cosa muchas veces.

Vamos a explicar un poco lo que hacen las órdenes que has introducido.

Las tres primeras órdenes nos inicializan el estado de la pantalla y el estado de la tortuga.

Las cinco siguientes colocan a la tortuga en la posición en donde voy a empezar a pintar la primera letra.

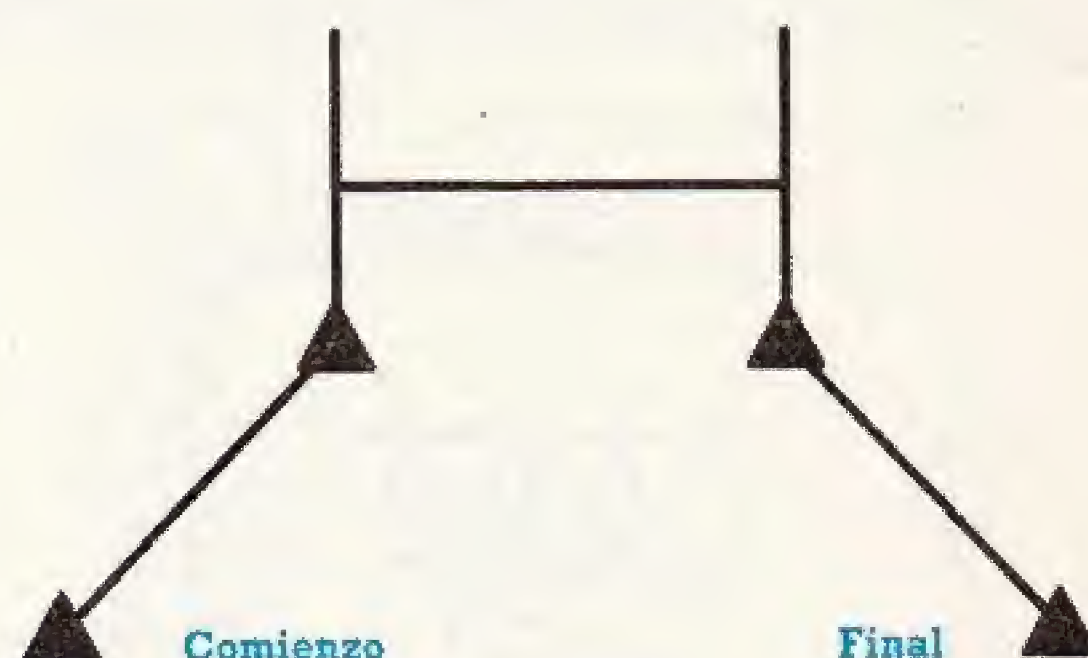
A continuación, pinto la H. Eso lo hago con las dos siguientes órdenes. Las cinco siguientes me colocan en la posición en donde quiero que se empiece a pintar la O. La orden siguiente pinta la O. Para situarme en donde quiero pintar la L, le introduzco las cuatro órdenes que siguen. La siguiente pinta la L. Lógicamente las cinco que hay después nos colocan en el comienzo de la última letra, y las dos siguientes pintarán la A.

No pienses que estas órdenes vas a tener que teclearlas siempre que quieras ver lo que hacen. Aprenderemos a almacenarlas para así poder disponer de ellas siempre que queramos.

Si quieres practicar, cambia los valores que hay y observa cómo varían las órdenes.

Por ejemplo, podrías cambiar el tamaño de las letras. Fíjate que en las órdenes con que dibujas las diferentes letras aparece un número que se repite en todas. Es 50. Esto significa que las letras son cuadradas y que tienen 50 puntos, tanto en horizontal como en vertical.

La tortuga comienza a dibujar cada letra desde el extremo inferior izquierdo de cada letra y al terminar se posiciona en el extremo inferior derecho.



Si varías el valor 50 variarás el tamaño de la letra, pero hazlo en todos los sitios donde aparezca 50; ya que si no las letras saldrán desproporcionadas.

Observa también que en las letras H y A hacemos un avance y un retroceso (25) que es justo la mitad del tamaño de nuestra letra (50). Si varías el tamaño no te olvides de variar también este valor, que tendrá que ser la mitad del nuevo valor que hayas dado al tamaño de las letras.

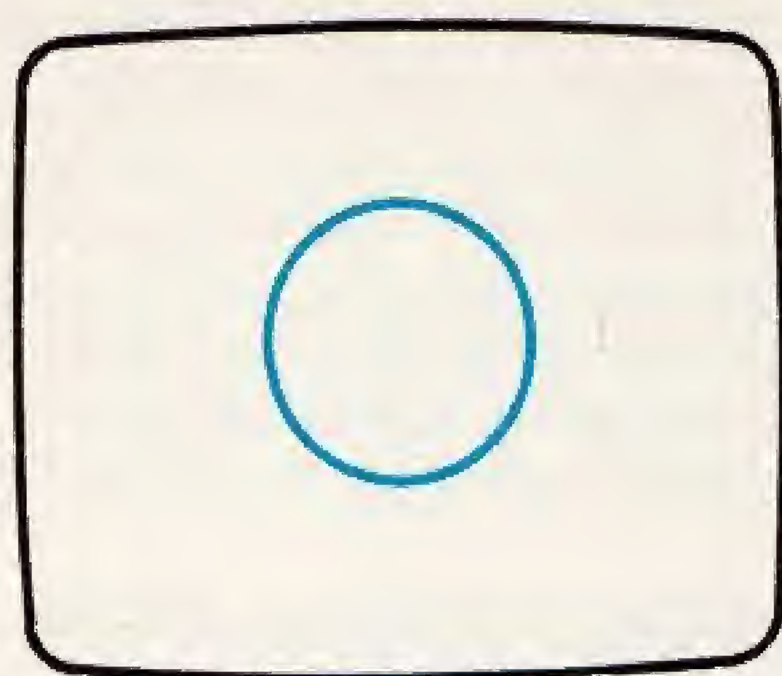
También puedes cambiar la separación entre las letras. Eso es muy fácil, ya que si observas las órdenes con las que desplazo a la tortuga hasta el comienzo de la siguiente letra, son las mismas en todas ellas.

En este caso avanzo 15 puntos que corresponden a la separación que le hemos dado. Varía este valor y variarás la separación entre las letras.

Bueno, ya has visto que con el **LOGO** se pueden dibujar palabras en el ordenador. Ya aprenderás cómo se hacen y podrás sorprender a tus amigos con saludos como éste.

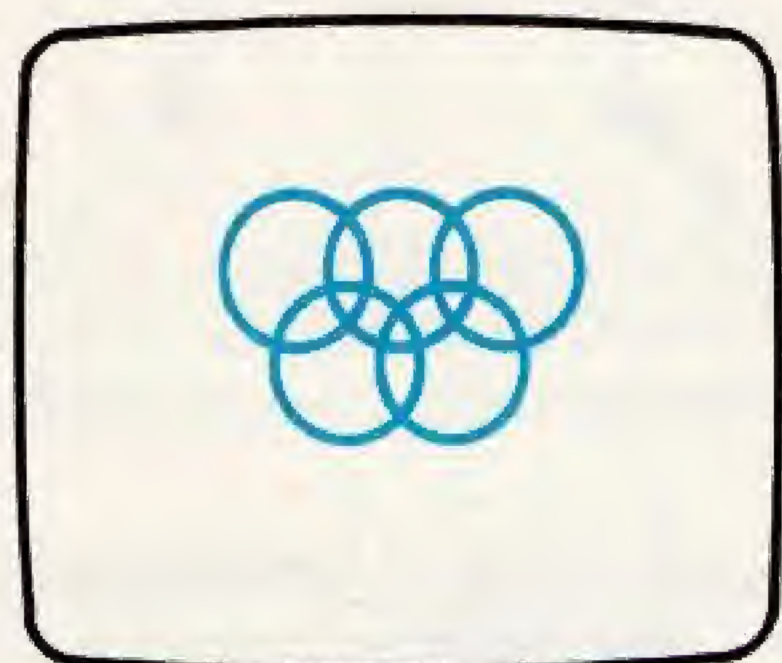
Veamos algo más:

No te olvides de dejar un espacio entre dos órdenes ni entre una orden y su argumento.



Fíjate que con el Logo también se pueden hacer círculos. Lo podemos hacer utilizando únicamente las órdenes que hemos usado hasta ahora.

Seguro que reconoces lo siguiente:



Estas son las órdenes para realizarlos:

AROS OLIMPICOS
SELECCION DE PANTALLA
Y ESTADO DE LA TORTUGA

```
? PM
? BP
? OT
? SL
```

CENTRANDO DIBUJO

```
? GI 90 AV 60
? GD 90 AV 50
```

DIBUJA LOS TRES AROS SUPERIORES

```
? BL
? REPITE 3[REPITE 36[AV 4 GD 10]SL
         GD 90 AV 35 GI 90 BL]
```

SITUO LA TORTUGA DONDE EMPIEZA
A DIBUJAR LOS DOS AROS INFERIORES

```
? SL
? RE 35 GI 90
? AV 87 GD 90
```

DIBUJA LOS DOS AROS INFERIORES

```
? BL
? REPITE 2[REPITE 36[AV 4 GD 90]SL
         GD 90 AV 35 GI 90 BL]
```

Como ves, con las órdenes que antes usábamos para pintar una palabra, ahora podemos hacer un dibujo a base de circunferencias.

Fíjate bien cómo hace este dibujo.

Lo primero que hacemos es situar a la tortuga en un sitio determinado para que el dibujo nos quede más o menos centrado.

Una vez colocada en el sitio deseado, dibujo los tres aros superiores.

Aquí lo único que se hace es repetir tres veces el mismo dibujo (una circunferencia) y cuando termino de dibujar la primera nos situamos en donde queremos empezar a dibujar la segunda. Igual ocurre entre la segunda y la tercera. Por eso, después de dar las órdenes para que dibuje un aro, le damos otras para que se posicione en el sitio idóneo.

Una vez que están dibujados los tres aros superiores, tenemos que posicionarnos en dónde queremos empezar a pintar los dos aros inferiores y simplemente repetir el proceso anterior, pero ahora únicamente dos veces.

CUADRO RESUMEN

Muestratortuga (MT)

Hace aparecer la tortuga en la posición en que se encuentra en un momento determinado.

Si esta orden se da nada más comenzar, la tortuga aparece en el centro de la pantalla.

Ocultatortuga (OT)

Hace desaparecer la tortuga en la posición en la que se encuentra.

Curioso, el Logo es el único lenguaje que te saluda cuando entra a trabajar.

Pantallamixta (PM)

Pone la pantalla en la modalidad mixta.

Divide la pantalla en dos zonas diferenciadas. Una exclusivamente para gráficos y otra para textos. En la zona gráfica aparecen todos los dibujos que se realicen y en la zona de texto escribiremos las órdenes y recibiremos los mensajes posibles que nos dé el ordenador.

Borrapantalla (BP)z

Borra la zona de la pantalla destinada a los gráficos y posiciona la tortuga en el centro.

Avanza n (AV n)

Avanza en la dirección en la que la tortuga está orientada el número de puntos que se especifican en "n".

Retrocede (RE n)

Retrocede en la dirección en la que la tortuga está orientada el número de puntos que se especifican en "n".

Giraderecha n (GD n)

Gira a la derecha, desde la posición en que está la tortuga, el número de grados (n) que se especifica.

Giraizquierda n (GI n)

Gira a la izquierda, desde la posición en que está la tortuga, el número de grados (n) que se especifica.

Bajalápiz (BL)

Activa el lápiz de la tortuga.

Desde el momento en que se dá esta orden todos los desplazamientos de la tortuga los hará dejando un rastro tras de sí.

Subelápiz (SL)

Desactiva el lápiz de la tortuga.

Desde que se da esta orden todo desplazamiento de la tortuga los hará sin dejar rastro.

¿Te atreves a realizar estos ejercicios?

1) *Dibujar con la tortuga un cuadrado, un triángulo y una circunferencia.*
2) *¿Qué cambiarías para dibujarlo sin la tortuga?*
3) *Dibuja esta escalera.*



4) *Dibuja un armario.*
5) *¿Son correctas las órdenes siguientes?*

```
BP
MUESTRA TORTUGA
AV50
GI 90
AVANZA 60
GDERECHA 90
AVANZA 3000
GD 360
```

Solución a los ejercicios

1. a) LADO = 50

```
CUADRADO
  INICIALIZACION
    ? PM
    ? SL
    ? BP
    ? MT

  CENTRANDO DIBUJO
    ? GD 45
    ? AV 25
    ? GD 135

  DIBUJO CUADRADO
    ? BL
    ? AV 50 GD 90
    ? AV 50 GD 90
    ? AV 50 GD 90
    ? AV 50 GD 90
```

Con el logo es muy fácil dibujar. Con otros te las ves y te las deseas.

O BIEN

DIBUJO CUADRADO
? BL
? REPITE 4[AV 50 GD 90]

b) LADO = 50

TRIANGULO
INICIALIZACION
? PM
? SL
? BP
? MT

CENTRANDO DIBUJO
? GD 90
? RE 25

DIBUJO TRIANGULO
? BL
? AV 50 GI 120
? AV 50 GI 120
? AV 50 GI 120

O BIEN

DIBUJO TRIANGULO
? BL
? REPETIR 3[AV 50 GI 120]

c) CIRCUNFERENCIA
INICIALIZACION

? PM
? SL
? BP
? MT

CENTRANDO DIBUJO
? GI 90
? AV 50
? GD 90

DIBUJO CIRCUNFERENCIA
? BL
? REPITE 36[AV 4 GD 10]

O BIEN

DIBUJO CIRCUNFERENCIA
? BL
? REPITE 36[AV 8 GD 10]

2. La orden que hace desaparecer la tortuga es:
OCULTATORTUGA (OT)

Si la introduces en lugar de MT en todos los dibujos, la tortuga no se verá.

3. ESCALERA
INICIALIZANDO
? PM
? SL
? BP
? OT
CENTRANDO DIBUJO
? GI 90
? AV 15
? GD 90
? AV 90
DIBUJO LATERALES
? BL
? RE 135
? GD 90
? SL
? AV 25
? GI 90
? BL
? AV 135
? RE 8
? GI 90
DIBUJO PELDAÑOS
? REPITE 9[AV 25 RE 25 GD 90 RE 14 GI 90]

4. ARMARIO
INICIALIZACION
? PM
? SL
? BP
? OT
CENTRANDO DIBUJO
? RE 50
? GI 90
DIBUJO ARMARIO
? BL
? AV 25 GD 90
? AV 125 GD 90
? AV 50 GD 90
? AV 125 GD 90
? AV 25 GD 90
DIBUJO PUERTAS
? SL
? AV 5
? BL
? GI 90 AV 20
? GI 90 AV 115
? GI 90 AV 40
? GI 90 AV 115
? GI 90 AV 20
? GI 90 AV 115

Puedes elegir trabajar viendo la tortuga o sin verla.

DANDO PROFUNDIDAD
? SL
? AV 5 GI 90
? BL
? AV 25 GD 135
? AV 15 GD 45
? AV 50 GD 135
? AV 15 Gi 45
? AV 125 GI 135
? AV 15 GI 45
? AV 125

DIBUJO TIRADORES
? SL
? CENTRO
? AV 10 GI 90
? AV 5 GD 90
? BL
? AV 10
? SL
? GD 90 AV 10
? GI 90 AV 1
? BL
? RE 10

5. BP: correcta.
MUESTRA TORTUGA: incorrecta. MUESTRA y TORTUGA tienen que ir juntas.
AV50: incorrecta. AV y 50 tienen que ir separadas.
GI 90: correcta.
AVANZA 30: correcta.
GDERECHA 90: no existe esta orden. La correcta es GIRADERECHA 90 o bien GD 90.
AVANZA 3000: correcta.
GD 360: correcta.

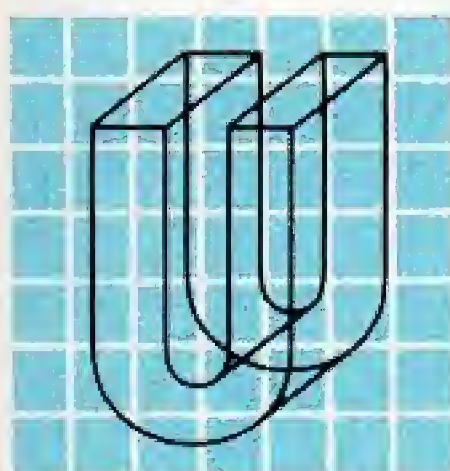
■ ***No te olvides que en los MSX tienes que introducir el cartucho del Logo con el ordenador apagado.***

■ ***En los MSX no tienes que dar ninguna orden para que el Logo se cargue. Lo hace automáticamente.***

■ ***En el MSX, cuando arranca el Logo y tecleamos MT, aparece un muñeco en vez del triángulo. Si quieres que aparezca el triángulo tienes que teclear la orden TORTUGA.***

■ ***En el Spectrum la orden SUBELAPIZ (SL) es SINLAPIZ (SL) y BAJALAPIZ (BL) es CONLAPIZ (CL).***

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS



NA de las características más importantes que tienen los ordenadores que actualmente se comercializan en el mercado es la capacidad que tienen de dibujar y realizar gráficos en colores de una forma rápida y sencilla. Es posible que el BOOM informático en el que nos encontramos inmersos no se hubiese dado si no tuviesen esta capacidad gráfica, pues se puede decir que un ordenador que dibuja deja de ser una fría máquina para pasar a ser un caluroso amigo.

Todo gráfico o dibujo que queramos realizar lo veremos plasmado en la pantalla de nuestro ordenador. Pero antes es necesario entender qué es, cómo es, qué características tiene y cómo se usa nuestra pantalla en la que trabajaremos como si de un lienzo de pintura se tratase.

Forma y dimensiones de la pantalla

La pantalla de nuestro ordenador está dividida en cuadraditos invisibles. En cada uno de ellos cabe una letra o un número. Estos cuadraditos están numerados para que el ordenador y nosotros podamos referirnos a ellos.

Nuestra pantalla está dividida en líneas, y cada línea tiene un cierto número de cuadraditos. Por ejemplo, en el SPECTRUM, tenemos 22 líneas con 32 cuadraditos en cada línea. Para referirnos a ellas, y para que el ordenador nos entienda, las numeraremos de la siguiente forma:

Las líneas del 0 al 21 y los cuadraditos o caracteres de cada línea del 0 al 31.

De esta manera nos podemos referir a cualquier parte de la pantalla con sólo decir su número de línea y el número de cuadradito dentro de dicha línea.

Ya hemos dicho cómo es la pantalla en el SPECTRUM (ver fig. 1). En el AMSTRAD tenemos tres modos distintos de pantalla, que numeraremos como 0, 1 y 2, con distintas dimensiones cada uno de ellos.

0 1 2 3 28 29 30 31
0 1 2 19 20 21 22 23

Fig. 1.—Aunque el Spectrum tiene 24 cintas las dos últimas están reservadas al ordenador.

En el modo 0, tenemos 25 líneas de 20 caracteres o cuadraditos en cada línea. En el modo 1 seguimos teniendo 25 líneas, pero esta vez con 40 caracteres por línea. Y en el modo 2 tenemos 80 caracteres por línea y las mismas 25 líneas.

La forma de numerar las líneas en el AMSTRAD es de 1 a 25 y los caracteres o cua-

draditos de 1 a 20, 40 u 80, según el modo de pantalla en el que nos encontremos.

Para ver cómo es cada modo de pantalla en el AMSTRAD, ejecuta el programa 1 y fíjate en la figura 2.

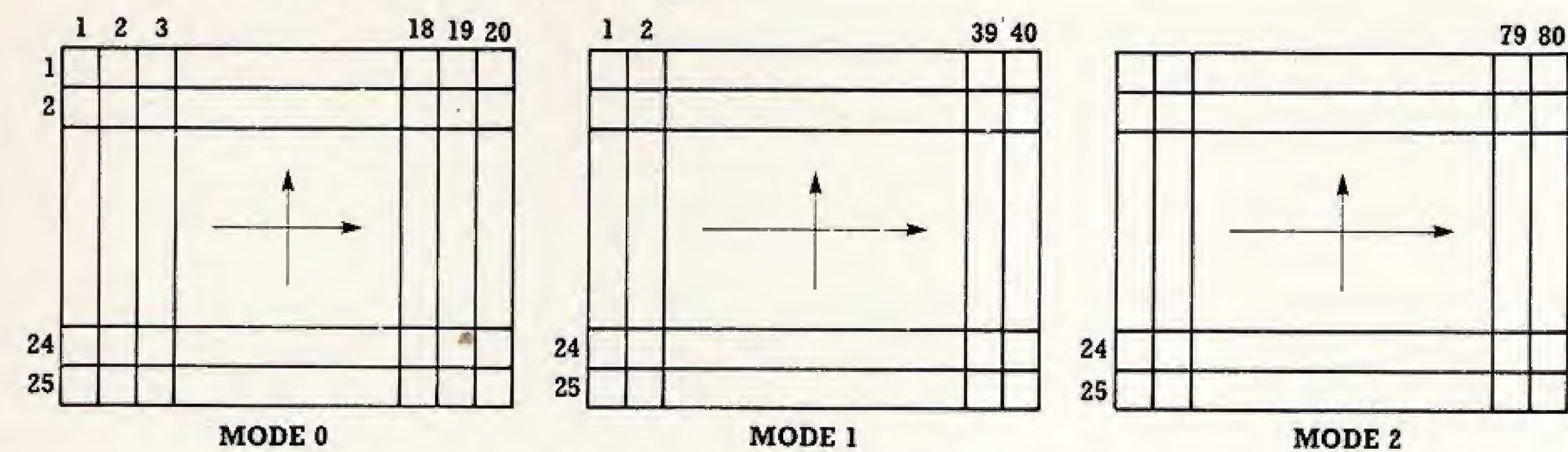


Fig. 2.—Los modos de pantalla en el Amstrad.

```
 10 REM ********************************
 20 REM * PROGRAMA PARA VER LOS MODOS *
 30 REM * DE PANTALLA EN EL AMSTRAD   *
 40 REM ********************************
 50 REM
 60 FOR I=1 TO 2
 70    MODE I
 80    PRINT "ESTAMOS EN EL MODO ";I;" DE
       PANTALLA."
 90    FOR J=1 TO 1000
100    NEXT J
110 NEXT I
120 END
```

Como podemos apreciar en el programa, la instrucción para cambiar de modo de pantalla es MODE i, "i" es el número de pantalla que queremos ver.

En el COMMODORE sólo hay un modo de pantalla. Tiene 25 líneas de 40 caracteres cada línea. Se numeran de 0 a 24 las líneas y de 0 a 39 las columnas.

En el IBM tenemos, al igual que en el AMSTRAD, tres tipos de pantalla distintos, aunque de momento sólo nos interesa el modo 0. En este modo la pantalla tiene 24 líneas y cada línea puede tener 40 u 80 caracteres. En el IBM la forma de cambiar la anchura de la pantalla es mediante la sentencia WIDTH n, donde "n" es 40 u 80, según queramos tener 40 u 80 caracteres por línea.

Para entrar en el modo O en el IBM, sólo tenemos que poner:

SCREEN 0

En el MSX también hay varios tipos de pantalla, pero sólo nos interesan de momento dos de ellos.

SCREEN 0. Pantalla de 24 líneas y de 40 caracteres por línea.

SCREEN 1. Pantalla de 24 líneas y de 32 caracteres por línea.

Su numeración, al igual que en el IBM, va de 1 a 24 líneas y de 1 a 32 (o 40) columnas.

Aparte de todo lo dicho, el MSX tiene también la sentencia WIDTH, pero la forma de actuar es distinta de la del IBM.

WIDTH fija el número máximo de columnas que vamos a utilizar en la pantalla. En el modo 0, se podrá variar el número de caracteres por línea entre 1 y 40, y en el modo 1 entre 1 y 32.

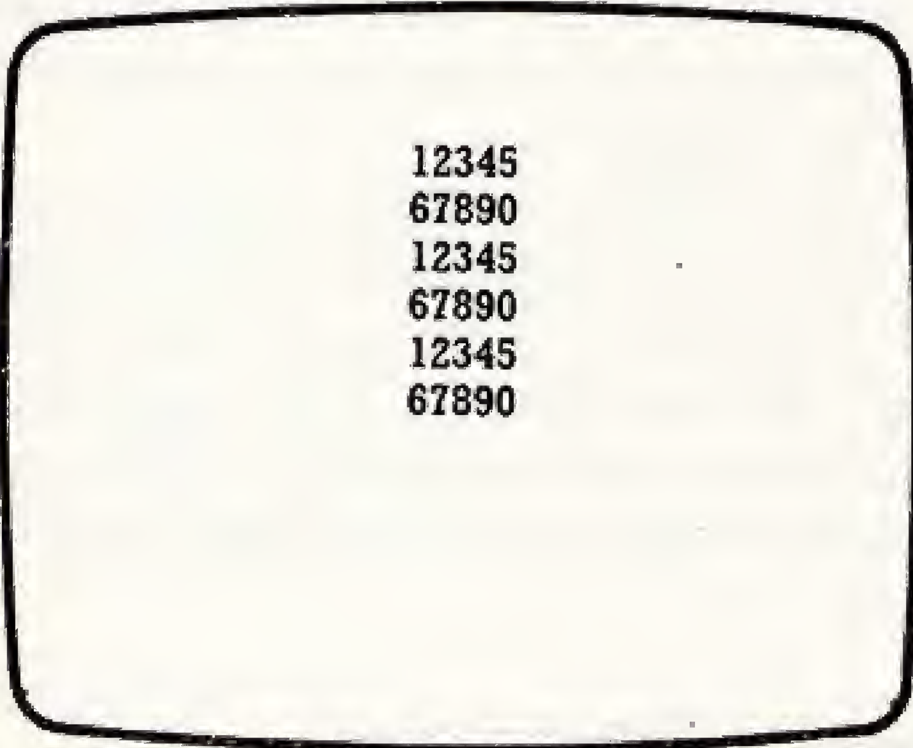


Fig. 3.—Así queda la pantalla en el MSX usando WIDTH 5.

Como ejemplo teclea el programa 2.

Para ilustrar mejor lo dicho hasta ahora, propongo el programa 3 para los usuarios del SPECTRUM y el 4 para todos los demás.

```
 10 REM *****************
 20 REM * USO DEL WIDTH *
 30 REM * PARA MSX      *
 40 REM *****************
 50 REM
 60 SCREEN 0
 70 FOR I=1 TO 40
 80    WIDTH I
 90    PRINT "123456789012345678901234567890"
100    FOR J=1 TO 1000
110    NEXT J
120 NEXT I
130 END
```

```
 10 REM ************************************
 20 REM * VISUALIZACION DE LOS CUADRADITOS *
 30 REM * EN EL SPECTRUM                   *
 40 REM ************************************
 50 REM
 60 FOR X=0 TO 255 STEP 8
 70     PLOT X,0
 80     DRAW 0,170
 90 NEXT X
100 FOR Y=0 TO 170 STEP 8
110     PLOT 0,Y
120     DRAW 255,0
130 NEXT I
140 OVER 1
150 LIST
160 OVER 0
170 STOP
```

```
 10 REM **********************************
 20 REM * VISUALIZACION DEL NUMERO DE    *
 30 REM * LINEAS Y DEL NUMERO DE CARAC-  *
 40 REM * TERES POR LINEA PARA TODOS     *
 50 REM * LOS ORDENADORES                *
 60 REM **********************************
 70 REM
 80 REM PONED VUESTRO ORDENADOR EN EL MODO
 90 REM DE 40 CARACTERES POR LINEA. EN EL
100 REM SPECTRUM CAMBIAR TODOS LOS 40 DEL
110 REM LISTADO POR 32.
120 REM
130 CLS:REM <-- EN EL COMMODORE CAMBIARLO
    POR: PRINT "<SHIFT- HOME>"
140 FOR I=1 TO 40
150     PRINT "*";
160 NEXT I
170 FOR I=2 TO 23
180     PRINT "*";TAB(40);"*"
190 NEXT I
200 FOR I=1 TO 40
210     PRINT "*";
220 NEXT I
230 END
```

Las zonas de la pantalla

Ahora ya estamos preparados para meternos de lleno a conocer el ordenador. Una de las cosas más importantes que hay que saber sobre la pantalla es que está dividida en zonas.

Una zona es una subdivisión en vertical de la pantalla. Así en el SPECTRUM tenemos la pantalla dividida en dos zonas. Una es la que va desde la columna del carácter número 0 hasta la 15, ambas inclusive. La otra va desde la columna 16 a la 31.

En el IBM las divisiones son cada 10 caracteres, o sea, van de la columna 1 a la 10, de la 11 a la 20, de la 21 a la 30, etc.

Para posicionarnos en una zona específica utilizaremos la coma (,) desde un PRINT, de la siguiente manera:

PRINT "1", "2", "3", "4"

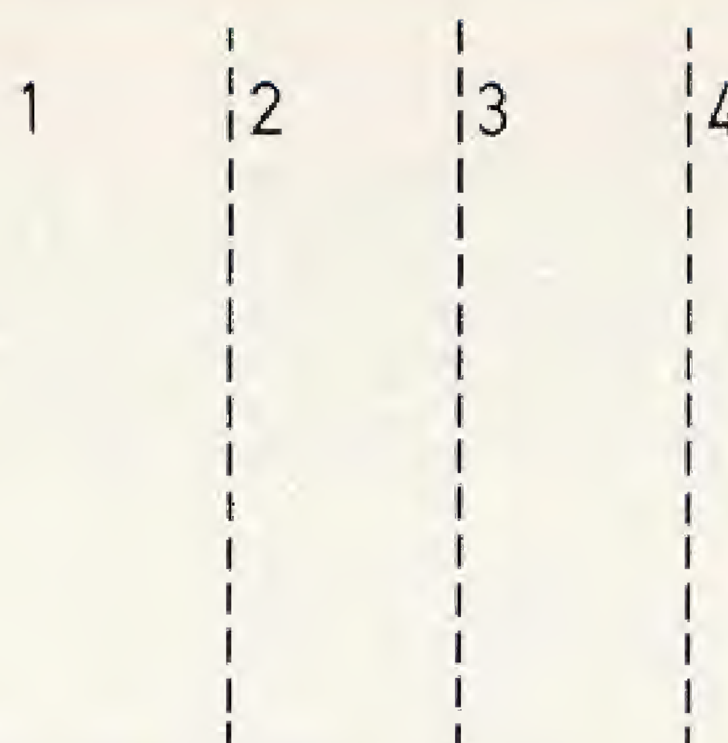


Fig. 4.—Las zonas de la pantalla.

```
 10 REM ****************************
 20 REM * LAS ZONAS DE LA PANTALLA *
 30 REM ****************************
 40 REM
 50 PRINT "1","2","3","4"
 60 PRINT "CADA NUMERO ESTA EN SU ZONA"
 70 PRINT
 80 PRINT ,,"3","4"
 90 PRINT "AHORA SOLO EN LAS DOS ULTIMAS ZONAS"
100 PRINT
110 PRINT "1",,,"4"
120 PRINT "AHORA SOLO EN LA PRIMERA Y EN LA
    ULTIMA"
130 PRINT
140 PRINT ,"2","3"
150 PRINT "Y AHORA LAS DOS DEL MEDIO"
160 PRINT
170 PRINT "1","2","3","4","5"
180 PRINT "QUE PASA AHORA?"
190 END
```

Nos aparecerá un uno (1) en el principio de la primera zona, un dos (2) en la segunda, un tres (3) en la tercera y un cuatro (4) en la cuarta. Para entender mejor todo esto fíjate en la figura 4 y ejecuta el programa 5.

El cursor, cómo posicionarlo

Hasta ahora hemos aprendido lo que es la pantalla del ordenador y su subdivisión en diferentes zonas, pero no hemos hablado para nada del cursor ni para qué sirve.

El cursor es ese pequeño cuadradito o guión parpadeante que se ve en el ordenador cuando lo acabamos de encender y que nos indica que está listo para empezar a trabajar. El cursor señala la posición donde se imprimirá el siguiente carácter que pulsemos en el teclado o que le mandemos escribir.

Cuando el ordenador está esperando que pulsemos alguna tecla, el cursor, como ya hemos dicho, es visible y está parpadeando, pero cuando nos encontramos ejecutando un

programa, aunque no lo veamos, el cursor sigue estando ahí.

Cuando en un programa le decimos al ordenador, que nos escriba algo en la pantalla (mediante la sentencia PRINT), el cursor, ya invisible, está diciéndole al ordenador dónde tendrá que escribir la próxima vez. Por ejemplo, si escribimos:

```
10 PRINT "HOLA"
20 GOTO 20
```

cuando hagamos RUN, el programa imprimirá la palabra HOLA y el cursor se quedará a la derecha de la A esperando otra sentencia PRINT para continuar escribiendo a partir de esa posición. Por eso si intercambiamos la línea:

```
15 PRINT "ADIOS"
```

lo que veremos en la pantalla al hacer RUN sería:

HOLAADIOS

y como después de las comillas que cierran la palabra ADIOS no pusimos un punto y coma (;) el cursor no se quedará a la derecha de la S, sino que estará en la línea siguiente.

Para lograr posicionar el cursor en cualquier parte de la pantalla desde dentro de un programa utilizaremos la función LOCATE, cuya sintaxis es:

LOCATE Y,X

donde Y es la fila a la que nos queremos mover y X es el carácter dentro de dicha línea. En el MSX los argumentos van al contrario, esto es, en vez de poner primero la Y y después la X se pone primero la X y después la Y.

LOCATE X,Y

Ni que decir tiene que los argumentos X e Y, aparte de tener que ser enteros, pueden ser números, variables o cualquier expresión que dé como resultado un número.

Si después de la sentencia LOCATE ponemos un PRINT, el mensaje aparecerá en la posición dada por X y por Y. Por ejemplo:

```
LOCATE 6,10: PRINT "AQUI ESTOY"
```

empezará a imprimir la frase "AQUI ESTOY" en la columna 10 de la fila 6 (en el MSX se pondría el mensaje en la columna 6 de la fila 10).

En el SPECTRUM no existe la sentencia LOCATE, pero se puede sustituir por PRINT AT Y,X obteniéndose el mismo resultado. Su sintaxis es:

```
PRINT AT Y,X; "mensaje a imprimir"
```

En el COMMODORE no hay ninguna función que realice nada parecido, pero utilizando la rutina que se da en este mismo fascículo en la sección TRUCOS DE PROGRAMACION se obtienen los mismos resultados.

A continuación aparece un programa de demostración de todo lo que hemos dado hasta ahora.

```
 10 REM *************************************
 20 REM * DEMOSTRACION DE LA SENTENCIA LOCATE *
 30 REM *************************************
 40 REM
 50 INPUT "En que fila imprimo el mensaje    ";Y
 60 INPUT "En que columna imprimo el mensaje";X
 70 INPUT "Que mensaje imprimo              ";M$
 80 REM
 90 LOCATE X,Y
100 PRINT M$
110 END

    Para el SPECTRUM quitar la linea 80 y
    sustituir la linea 70 por:

 70 PRINT AT Y,X;M$

    Para el COMMODORE sustituir la linea 70 por:

 70 GOSUB 9950
```

El código ASCII

Desde el principio de este fascículo estamos hablando de los caracteres. Pero ¿qué es un carácter?

Se denominan caracteres a todas las letras, números, signos de puntuación y, en general, a cada uno de los 256 caracteres que nuestro ordenador tiene almacenados en su memoria ROM (memoria de solo lectura).

El ordenador asigna un número a cada carácter. Esto es así para que él, y nosotros, podamos referirnos a cualquiera de ellos por su número, pues el ordenador no entiende de otra cosa que no sean números. La numeración de los caracteres es más o menos estándar. A la serie de números que relacionan una serie de caracteres se le llama CODIGO. Al código que relaciona estos números con los caracteres en los microordenadores se le denomina CODIGO ASCII.

De los 256 caracteres del CODIGO ASCII, los 32 primeros (del 0 al 31) son caracteres de control (ya veremos qué significa esto) y el resto son caracteres que todos reconoceremos.

El programa 7 nos muestra todos los caracteres que tienen un CODIGO ASCII comprendido entre 32 y 255.

```
10 REM ******************************************
20 REM * VISUALIZACION DE LOS CARACTERS ASCII *
30 REM * DEL 32 AL 255                          *
40 REM ******************************************
50 REM
60 FOR I=32 TO 255
70    PRINT I ; CHR$(I)
80 NEXT I
90 END
```

Como puede verse dicho programa, cuando queramos imprimir un carácter del cual sólo conocemos su CODIGO ASCII, utilizaremos la función del BASIC:

PRINT CHR$(n)

donde n es el CODIGO ASCII de dicho carácter.

Hay otra función en BASIC que hace lo contrario. O sea, si conocemos el carácter, nos da su CODIGO ASCII.

LET A = ASC("A") o PRINT ASC("A")

Esta instrucción nos devolvería en A el número 65, que es el código ASCII de la letra A. Se puede ver que la letra ha de ir entre comillas. Si el carácter a comprobar estuviese almacenado en una variable alfanumérica la sintaxis sería:

LET A=ASC(B$) o PRINT ASC(B$)

sin poner la variable entre comillas. Si la variable alfanumérica constase de más de un carácter, la función ASC nos da el código del primer carácter de la cadena.

LET B$="HOLA": PRINT ASC(B$)

el ordenador imprimirá un 72, que es el código ASCII de la letra 'H'.

El programa 8 nos pide un mensaje y nos da el código ASCII de todas las letras de dicho mensaje.

```
 10 REM ****************************************
 20 REM * IMPRIME LOS CODIGOS ASCII DE TODOS *
 30 REM * LOS CARACTERES DE UN MENSAJE         *
 40 REM ****************************************
 50 REM
 60 INPUT "MENSAJE = ";M$
 70 FOR I=1 TO LEN(M$)
 80    PRINT ASC(MID$(M$,I,1));
 90    PRINT " ES EL CODIGO ASCII DE LA ";
100    PRINT MID$(M$,I,1)
110 NEXT I
120 END
```

Caracteres de control

Un poco más arriba hemos comentado que de los 256 caracteres que tiene nuestro ordenador, los 32 primeros son CARACTERES DE CONTROL. ¿Qué significa esto?

Hay una serie de funciones en nuestro ordenador que, aunque a nosotros no nos lo parezca, son caracteres. Entre ellos se encuentran las funciones de:

— Borrar pantalla.
— Mover los cursores.
— Hacer que suene un ruidito.
— DELETE (borrar el carácter que está a la izquierda del cursor).
— INSERT (insertar una letra o más entre dos que están juntas).
— Saltar de línea.

etcétera.

El carácter <borrar pantalla> (CHR$ (12)) tiene el mismo efecto que el comando CLS. En último extremo, CLS es un PRINT CHR$(12). Pruébalo.

El carácter que hace un ruidito es el número ocho (CHR$(7)) y hace lo mismo que el comando BEEP. Prueba a escribir PRINT CHR$(7).

Otro carácter de control es el trece (CHR$(13)), que hace que el cursor baje al principio de la línea siguiente. A este carácter se le llama RETORNO DE CARRO, por su parecido a lo que hace una máquina de escribir. Este carácter es el que lee el ordenador cuando pulsamos la tecla RETURN o ENTER (según qué ordenador).

Los caracteres de control suelen variar mucho de unos ordenadores a otros. Por ello, te recomendamos que mires tu Manual para poder conocer cuáles son los que tienes y qué efecto hacen. De todas maneras, aquí se incluye un cuadro con los caracteres de control más comunes y sus códigos ASCII (ver figura 9).

Almacenamiento de caracteres

Lógicamente el ordenador tiene que tener almacenados en algún lugar de su memo-



Fig. 5.

ria todos los caracteres. Pero tiene que guardar no sólo el número, sino también la forma. Para ello utiliza, por así decirlo, un papel cuadriculado en el que dibujará cada uno de los caracteres.

Como todos sabemos, la unidad más pequeña de información que puede tratar el ordenador es el BIT (Binary digIT). En los microordenadores que se comercializan estos BITs se agrupan de ocho en ocho formando lo que se llama BYTE (ver fig. 5). Si ponemos unos BYTES encima de otros y nos imaginamos que cada BYTE está formado por ocho cuadraditos, esto nos dará la rejilla que utiliza el ordenador para almacenar los caracteres (ver fig. 6).

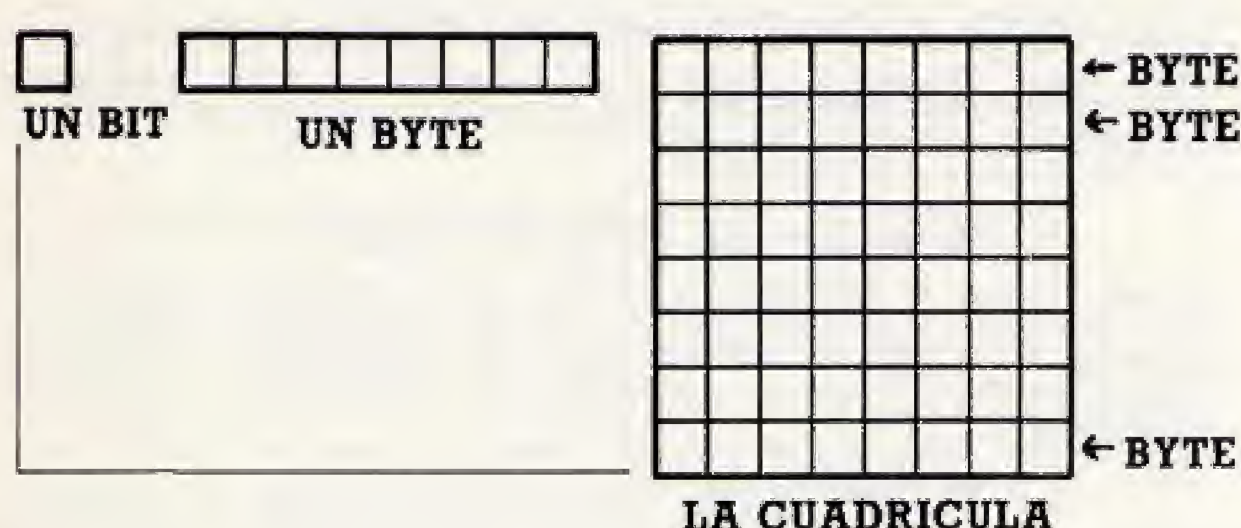


Fig. 6.

Si cada cuadradito fuese una bombilla, ésta podría estar de dos maneras: encendida o apagada. Si decimos que cada cuadradito es un dígito, éste puede tomar el valor cero (apagado) o uno (encendido). Cuando juntamos ocho BITs, formado un BYTE, cualquiera de ellos puede tomar el valor cero o el uno. Por ejemplo, un grupo de ocho BITs podría ser:

10010111

Que da la casualidad que es un número en base dos. Este número en base diez sería el 151.

Después de esto podemos decir que el ordenador sólo puede trabajar con números que estén en base dos y que no excedan de ocho dígitos, con lo que todo este razonamiento se esclarece.

Según esto, para almacenar la forma que tiene la letra A, el ordenador utiliza ocho BYTES (no BITs) uno encima de otro y enciende (pone a uno) los dígitos que dan la forma de la A (ver figs. 7 y 8).

Si quieres ver cómo están definidos los caracteres que tiene el AMSTRAD fíjate en la figura 9.

Ya sabemos cómo almacena el ordenador una letra, pero ¿cómo almacena el grupo entero de letras? El primer carácter que tiene

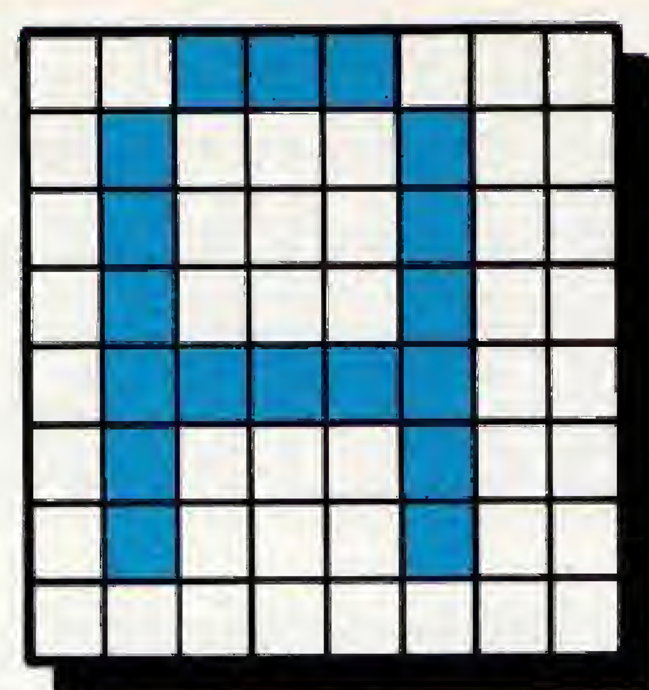


Fig. 7.

0	0	0	1	1	1	0	0	→	28
0	0	1	0	0	0	1	0	→	34
0	0	1	0	0	0	1	0	→	34
0	0	1	0	0	0	1	0	→	34
0	0	1	1	1	1	1	0	→	52
0	0	1	0	0	0	1	0	→	34
0	0	1	0	0	0	1	0	→	34
0	0	0	0	0	0	0	0	→	0

Fig. 8.

almacenado es el carácter 32 (el espacio), que se compone de ocho líneas de ocho ceros. A continuación va el carácter 33 (!) después el 34 (") y así hasta llegar al carácter 255.

Como vimos en el programa 7, de los 256 caracteres algunos son letras, números y caracteres de puntuación, pero hay muchos que son extraños y desconocidos. Este grupo suele incluir algunos caracteres griegos, pero los que a nosotros nos importan son los caracteres SEMIGRAFICOS. Estos caracteres son líneas verticales y horizontales, esquinas, cuadrados, triángulos, etc.

Con los caracteres semigráficos podemos, en cierta manera, dibujar cuadrados y otras cosas para hacer más bonitos nuestros programas. Un ejemplo lo podéis ver en los programas 9 y 10.

Si quieres pasar este programa a tu ordenador ten en cuenta que los caracteres y su código ASCII que se utilizan son los siguientes (ver fig. 10):

Para conocer mejor los caracteres semigráficos os recomiendo que practiquéis con ellos e intentéis introducirlos en vuestros programas.

Los ordenadores servirían de poco si no nos permitiesen definir nuestros propios caracteres. Con los caracteres que nosotros nos definamos podemos hacer juegos de marcianitos, escribir con letras que tengan acento,

CARACTER	CHR$	CARACTER	CHR$	CARACTER	CHR$	CARACTER	CHR$
[illegible]	0		32	@	64	`	96
[illegible]	1	!	33	A	65	a	97
⊥	2	"	34	B	66	b	98
[illegible]	3	#	35	C	67	c	99
[illegible]	4	$	36	D	68	d	100
⊠	5	%	37	E	69	e	101
✓	6	&	38	F	70	f	102
[illegible]	7	'	39	G	71	g	103
←	8	(	40	H	72	h	104
→	9	)	41	I	73	i	105
↓	10	*	42	J	74	j	106
↑	11	+	43	K	75	k	107
[illegible]	12	,	44	L	76	l	108
↵	13	-	45	M	77	m	109
[illegible]	14	.	46	N	78	n	110
[illegible]	15	/	47	O	79	o	111
[illegible]	16	0	48	P	80	p	112
[illegible]	17	1	49	Q	81	q	113
[illegible]	18	2	50	R	82	r	114
[illegible]	19	3	51	S	83	s	115
[illegible]	20	4	52	T	84	t	116
[illegible]	21	5	53	U	85	u	117
Π	22	6	54	V	86	v	118
[illegible]	23	7	55	W	87	w	119
X	24	8	56	X	88	x	120
[illegible]	25	9	57	Y	89	y	121
[illegible]	26	:	58	Z	90	z	122
[illegible]	27	;	59	[	91	{	123
[illegible]	28	<	60	\	92	¦	124
[illegible]	29	=	61	]	93	}	125
[illegible]	30	>	62	↑	94	~	126
[illegible]	31	?	63	_	95	[illegible]	127
[illegible]	128	^	160	/	192	[illegible]	224
[illegible]	129	´	161	\	193	[illegible]	225
[illegible]	130	¨	162	/	194	♣	226
[illegible]	131	£	163	^	195	♦	227
[illegible]	132	©	164	>	196	♥	228
[illegible]	133	¶	165	<	197	♠	229
[illegible]	134	§	166	∨	198	○	230
[illegible]	135	`	167	<	199	●	231
[illegible]	136	¼	168	[illegible]	200	□	232
[illegible]	137	½	169	◇	201	■	233
[illegible]	138	¾	170	X	202	♂	234
[illegible]	139	±	171	╱	203	♀	235
[illegible]	140	÷	172	╲	204	♩	236
[illegible]	141	¬	173	[illegible]	205	♪	237
[illegible]	142	¿	174	[illegible]	206	[illegible]	238
[illegible]	143	¡	175	[illegible]	207	[illegible]	239
[illegible]	144	α	176	|	208	↑	240
[illegible]	145	β	177	[illegible]	209	↓	241
[illegible]	146	γ	178	[illegible]	210	←	242
[illegible]	147	δ	179	[illegible]	211	→	243
[illegible]	148	ε	180	[illegible]	212	▲	244
[illegible]	149	θ	181	[illegible]	213	▼	245
[illegible]	150	λ	182	[illegible]	214	►	246
[illegible]	151	μ	183	[illegible]	215	◄	247
[illegible]	152	π	184	[illegible]	216	[illegible]	248
[illegible]	153	σ	185	[illegible]	217	[illegible]	249
[illegible]	154	φ	186	[illegible]	218	[illegible]	250
[illegible]	155	ψ	187	[illegible]	219	[illegible]	251
[illegible]	156	χ	188	[illegible]	220	[illegible]	252
[illegible]	157	ω	189	[illegible]	221	[illegible]	253
[illegible]	158	Σ	190	[illegible]	222	[illegible]	254
[illegible]	159	Ω	191	[illegible]	223	[illegible]	255

```
 10 REM ******************************************
 20 REM * CUADRADO CON CARACTERES SEMIGRAFICOS *
 30 REM * PARA EL COMMODORE                     *
 40 REM ******************************************
 50 REM
 60 PRINT "<SHIFT-HOME>"
 70 PRINT CHR$(176);
 80 FOR I=1 TO 10
 90     PRINT CHR$(99);
100 NEXT I
110 PRINT CHR$(174)
120 FOR I=1 TO 10
130     PRINT CHR$(98);"                    ";CHR$(98)
140 NEXT I
150 PRINT CHR$(173);
160 FOR I=1 TO 10
170     PRINT CHR$(99);
180 NEXT I
190 PRINT CHR$(189);
200 END
```

```
 10 REM ******************************************
 20 REM * CUADRADO CON CARACTERES SEMIGRAFICOS *
 30 REM * PARA EL IBM                           *
 40 REM ******************************************
 50 REM
 60 CLS
 70 PRINT CHR$(201);
 80 FOR I=1 TO 10
 90     PRINT CHR$(205);
100 NEXT I
110 PRINT CHR$(187)
120 FOR I=1 TO 10
130     PRINT CHR$(186);"                    ";CHR$(186)
140 NEXT I
150 PRINT CHR$(200);
160 FOR I=1 TO 10
170     PRINT CHR$(205);
180 NEXT I
190 PRINT CHR$(188)
200 END
```

escribir en griego o en árabe, hacer que las letras de nuestro ordenador se parezcan a las que se utilizan en las películas del Oeste o incluso escribir en un alfabeto inventado por nosotros.

Para cambiar los caracteres que tiene el ordenador sólo tendremos que poner en la zona de la memoria donde éstos se almacenan la configuración de nuestro propio juego de caracteres. Antes de explicar cómo hacerlo necesitamos saber algunas cosas.

En la memoria del ordenador sólo hay números. Números que, aunque nosotros no entendamos lo que significan, el ordenador sí entiende.

Por otra parte, la memoria del ordenador está dividida en dos partes: la ROM (memoria de sólo lectura) y la RAM (memoria de lectura y escritura).

En la ROM no podemos escribir, pero podemos mirar lo que ya está escrito. En la RAM podemos escribir lo que queramos y mirar lo que ya está escrito.

Pero ¿cómo mirar lo que hay en una po-

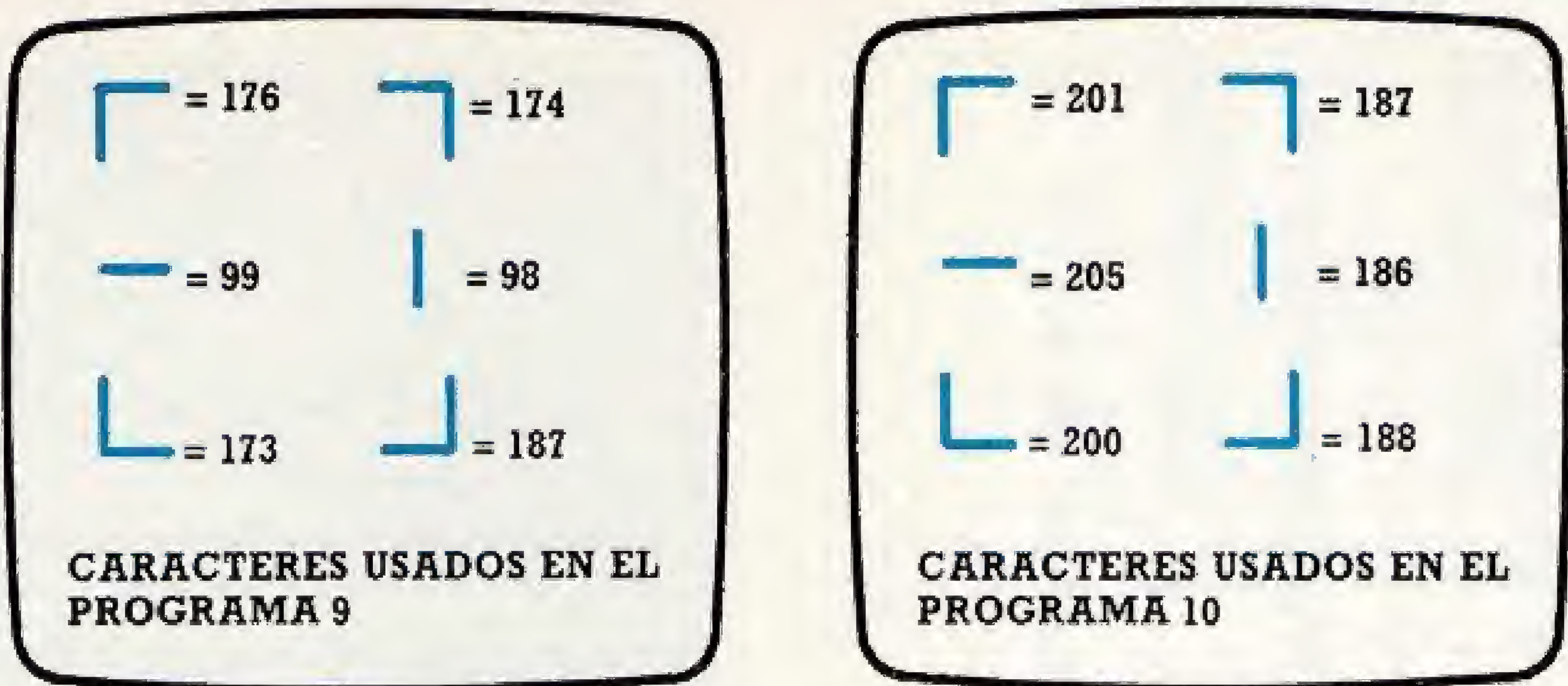


Fig. 10—Fíjate en estos caracteres y búscalos en el código ASCII de tu ordenador. Cuando los encuentres sustituye los números en el programa 10.

sición de memoria? Para hacerlo se utiliza un comando del BASIC, el comando PEEK. Este nos devuelve el número que hay escrito en una posición de memoria. Su sintaxis es:

PEEK (p)

donde 'p' es la posición de memoria que queremos leer. Para ver un ejemplo ejecuta el programa 11.

```
 10 REM ******************************
 20 REM * PROGRAMA PARA VER QUE HAY *
 30 REM * DENTRO DE LA MEMORIA       *
 40 REM ******************************
 50 REM
 60 INPUT "POSICION DE MEMORIA INICIAL ";N1
 70 INPUT "POSICION DE MEMORIA FINAL   ";N2
 80 FOR I=N1 TO N2
 90    PRINT "DIRECCION = ";I," VALOR =";PEEK(I)
100 NEXT I
110 END
```

La forma de escribir en una posición de memoria es con el comando POKE, cuya sintaxis es:

POKE p,n

donde 'p' es la posición de memoria donde queremos escribir y 'n' el valor que queremos escribir. El valor de 'n' tiene que estar comprendido entre 0 y 255, ya que el ordenador no entiende números mayores que 255 ni menores que 0. Aunque esto último te parezca mentira, es verdad. Cuando tú le dices al ordenador:

PRINT 12000

el ordenador te imprime 12000, y 12000 es mayor que 255. La verdad que el ordenador realiza una serie de trucos antes de imprimir dicho número, lo cual no significa que pueda almacenarlo en un solo BYTE.

A la hora de utilizar el comando POKE tienes que tener mucho cuidado, pues puedes meter valores en zonas de la memoria reservadas para uso del ordenador. En estas zonas el ordenador almacena ciertos datos como la hora, el día o la memoria libre. Si tú tocas estas zonas de memoria el ordenador se puede volver loco, pero no te preocupes, si lo apagas y lo vuelves a encender se volverá normal otra vez.

Te aconsejo que practiques con estos dos últimos comandos para acostumbrarte a ellos.

Ahora ya estamos en condiciones de definir nuestros propios caracteres gráficos. Antes de ponernos delante del ordenador, tenemos que saber qué nuevos caracteres queremos realizar. Para ello cogemos un lápiz y un papel cuadriculado y dibujamos en el papel, cuadrado a cuadrado, el nuevo carácter. No se te olvide que tiene que tener 8 puntos de alto y otros ocho de ancho (8 × 8 = 64 cuadraditos).

Una vez dibujado nuestro carácter cogemos cada fila y la escribimos como una tira de unos y ceros. A continuación pasamos dicho número, que está en base dos, a base diez y continuamos con la siguiente línea hasta llegar a la octava.

Cuando terminemos tenemos que tener ocho números en base diez. Pero ¿qué hacemos con estos ocho números? En el fascículo siguiente veremos cómo almacenarlos y cómo hacer cosas con ellos.

A continuación tenemos un juego gráfico para que veas lo que podrás hacer con el tiempo tú y el ordenador. Esto será cuando hayas aprendido lo suficiente, que será muy pronto (ver figs. 11 y 12).

```
Ok
PUNTOS = 0                  VIDAS = 3
================================
+                                  +
+                                  +
+                                  +
+                                  +
+                                  +
+          -#-                     +
+                                  +
+          .                       +
+                !                 +
+                                  +
+                                  +
+                                  +
+                                  +
+                                  +
+                                  +
+  /------Ñ          /------Ñ      +
+  !======!          !======!      +
+             <=>                  +
================================
```

Fig. 11.—Aspecto de la pantalla del programa Juego-1.



Fig. 12.—Con el tiempo podrás realizar programas con unos gráficos tan bonitos como estos e incluso mejores.

```
PROGRAMA JUEGO-1
================
10 REM ************************
20 REM * JUEGO DE MARCIANITOS *
30 REM ************************
40 REM
50 REM (c) Ediciones Siglo Cultural, 1986.
60 REM Por Fco. Morales Guerrero.
70 REM
80 REM *** INICIALIZACION DEL JUEGO ***
90 REM
91 LET SW=0:LET SE=0:LET SA=0:LET SM=0
100 LET V=3              : REM  NUMERO DE VIDAS
110 LET PN=0             : REM  LOS PUNTOS
120 LET N$=" <=> "       : REM  NUESTRA NAVE
130 DIM E$(3)            : REM  TRES TIPOS DE NAVES ENEMIGAS DISTINTAS
140 LET E$(1)="-#-"      : REM  ENEMIGOS No. 1
150 LET E$(2)="/=Ñ"      : REM  ENEMIFOS No. 2
160 LET E$(3)="-.-"      : REM  ENEMIGOS No. 3
170 LET B$="!"           : REM  NUESTRO LASER
180 LET V$="."           : REM  EL LASER DE LOS ENEMIGOS
190 LET R$="/------Ñ"    : REM  TECHO DEL REFUGIO
200 LET S$="!======!"    : REM  CUERPO DEL REFUGIO
210 REM
220 REM *** EMPIEZA EL PROGRAMA ***
230 REM
240 GOSUB 1300           : REM  INSTRUCCIONES
250 REM
260 REM *** DIBUJO DE LA PANTALLA ***
270 REM
280 CLS
290 PRINT "PUNTOS = 0                  VIDAS = 3"
300 PRINT "================================"
310 FOR I=3 TO 17:PRINT "+                                  +":NEXT I
320 PRINT "+  ";R$;"          ";R$;"   +"
330 PRINT "+  ";S$;"          ";S$;"   +"
340 PRINT "+                                  +"
350 PRINT "================================";
360 REM
370 REM *** PROGRAMA PRINCIPAL ***
380 REM
390 LOCATE 20,15:PRINT N$:LET XN=15
400 LET NE=INT(RND*3+1):LET XE=INT(RND*28)+2:LET YE=INT(RND*17)+3:LET X1=XE:LET
Y1=YE
410 GOSUB 580
420 GOSUB 910:GOSUB 1000:GOSUB 1100
430 IF SA=1 THEN LET SA=0:GOSUB 670
440 IF SM=1 THEN LET SM=0:GOSUB 800:GOTO 400
450 GOTO 410
460 REM
```

```
470 REM *****************
480 REM * FIN DEL JUEGO *
490 REM *****************
500 REM
510 PRINT "                                "
520 PRINT " ****************************** "
530 PRINT " *        FIN DEL JUEGO       * "
540 PRINT " *       VIDAS  AGOTADAS      * "
550 PRINT " ****************************** "
560 LOCATE 21,1:END
570 REM
580 REM *** MOVIMIENTO DEL ENEMIGO ***
590 REM
600 IF RND>.8 AND SE=0 THEN LET SE=1:RETURN
610 LOCATE Y1,X1:PRINT "   ":LOCATE YE,XE:PRINT E$(NE):LET X1=XE:LET Y1=YE
620 LET XE=XE-(RND>.5 AND RND>.5)+(XE=28)
630 LET XE=XE+(RND>.5 AND RND<.5)-(XE=2)
640 LET YE=YE-(RND>.5 AND RND<.5)+(YE=17)
650 LET YE=YE+(RND<.5 AND RND>.5)-(YE=3)
660 RETURN
670 REM
680 REM *** ESTALLIDO DE LA NAVE ***
690 REM
700 FOR W=1 TO 3
710     LOCATE 20,XN:PRINT " <=>":GOSUB 1260
720     LOCATE 20,XN:PRINT "  <>":GOSUB 1260
730     LOCATE 20,XN:PRINT " <=>":GOSUB 1260
740     LOCATE 20,XN:PRINT " <> ":GOSUB 1260
750 NEXT W
760 LOCATE 20,XN:PRINT "    "
770 LET XN=15:LOCATE 20,XN:PRINT N$
780 LET V=V-1:GOSUB 1210
790 RETURN
800 REM
810 REM *** ESTALLIDO DE LA NAVE ENEMIGA ***
820 REM
830 FOR W=1 TO 3
840     LOCATE YE,XE:PRINT E$(NE):GOSUB 1260
850     LOCATE YE,XE:PRINT "...":GOSUB 1260
860 NEXT W
870 LOCATE YE,XE:PRINT "   "
880 LOCATE Y1,X1:PRINT "   "
890 LET PN=PN+10*NE:GOSUB 1230
900 RETURN
910 REM
920 REM *** CONTROL DEL TECLADO Y MOVIMIENTO DE LA NAVE ***
930 REM
940 REM
950 A$=INKEY$:IF A$="" THEN RETURN
960 IF A$="Q" THEN LET XN=XN-1-(XN=2):LOCATE 20,XN:PRINT N$:RETURN
970 IF A$="W" THEN LET XN=XN+1+(XN=27):LOCATE 20,XN:PRINT N$:RETURN
980 IF A$="P" AND SW=0 THEN LET SW=1:RETURN
990 RETURN
1000 REM
1010 REM *** DISPARO ***
1020 REM
1030 IF SW=0 THEN RETURN
1040 IF SW=1 THEN LET XD=XN+2:LET CC=18:LET SW=-1
1050 LOCATE CC+1,XD:PRINT " ":LOCATE CC,XD:PRINT B$
1060 IF (XD>XE-1 AND XD<XE+3) AND (CC=YE) THEN LET SM=1:BEEP
1070 LET CC=CC-1
1080 IF CC=2 THEN LET SW=0:LOCATE CC+1,XD:PRINT " "
1090 RETURN
1100 REM
1110 REM *** DISPARO DEL ENEMIGO ***
1120 REM
1130 IF SE=0 THEN RETURN
1140 IF SE=1 THEN LET XR=XE+1:LET CD=YE+1:LET SE=-1
1150 LOCATE CD-1,XR:PRINT " ":LOCATE CD,XR:PRINT V$:LET CD=CD+1
1160 IF CD=21 THEN LET SE=0:LOCATE CD-1,XR:PRINT " ":IF XR>XN AND XR<XN+4 THEN L
ET SA=1
1170 RETURN
1180 REM
1190 REM *** MARCADOR DE PUNTOS ***
1200 REM
1210 LOCATE 1,31:PRINT V:IF V=0 THEN GOTO 510
1220 RETURN
1230 REM
1240 LOCATE 1,9:PRINT PN:RETURN
1250 REM
1260 REM *** RETARDO ***
1270 REM
1280 FOR GG=1 TO 100:NEXT GG:RETURN
```

```
1290 REM
1300 REM *****************
1310 REM * INSTRUCCIONES *
1320 REM *****************
1330 REM
1340 CLS
1350 PRINT "*** ATAQUE LUNAR ***":PRINT
1360 PRINT "Por Fco. Morales Guerrero.":PRINT
1370 PRINT "INSTRUCCIONES":PRINT "-------------":PRINT
1380 PRINT "  Eres el comandante de las tres"
1390 PRINT "ultimas naves del universo."
1400 PRINT "  Los extraterrestres han con-"
1410 PRINT "quistado todo el universo. Todos"
1420 PRINT "los hombres estan esclavizados."
1430 PRINT "  Tu deber es matar a todos los"
1440 PRINT "extraterrestres antes de que "
1450 PRINT "ellos te maten a ti.":PRINT
1460 PRINT "  Para ello dispones de un po-"
1470 PRINT "tente laser de tecnologia japo-"
1480 PRINT "nesa.":PRINT:PRINT
1490 PRINT "PULSA UNA TECLA":GOSUB 1640
1500 CLS
1510 PRINT "Tu nave es : ";N$:PRINT
1520 PRINT "las de tus enemigos son:":PRINT
1530 PRINT E$(1);" ";E$(2);" ";E$(3):PRINT
1540 PRINT "Los bunkers protectores son:":PRINT
1550 PRINT R$:PRINT S$:PRINT
1560 PRINT "Utiliza las sigientes teclas":PRINT
1570 PRINT "Q = izquierda"
1580 PRINT "W = derecha"
1590 PRINT "P = disparo":PRINT
1600 PRINT "QUE TENGAS SUERTE !!!":PRINT
1610 PRINT "PULSA UNA TECLA":GOSUB 1640
1620 RETURN
1630 REM
1640 REM *******************
1650 REM * PULSA UNA TECLA *
1660 REM *******************
1670 REM
1680 LET A$=INKEY$:IF A$="" THEN GOTO 1680
1690 RETURN
```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

MODIFICACIONES DEL PROGRAMA JUEGO-1

Commodore

- En las líneas 280, 1340 y 1500, sustituir el CLS por: PRINT "<SHIFT-HOME>"
- Quitar el BEEP de la línea 1060.
- En las líneas donde ponga LET A$=INKEY$ sustituirlo por: GET A$.

Amstrad

- En todos los lugares donde ponga RND, sustituirlo por: RND(1).

MSX

- Donde ponga RND, sustituirlo por: RND(1).
- Recuerda que en el MSX los parámetros de la sentencia LOCATE van al contrario. Si ves que pone: LOCATE 20,XN, tendrás que poner LOCATE XN,20.

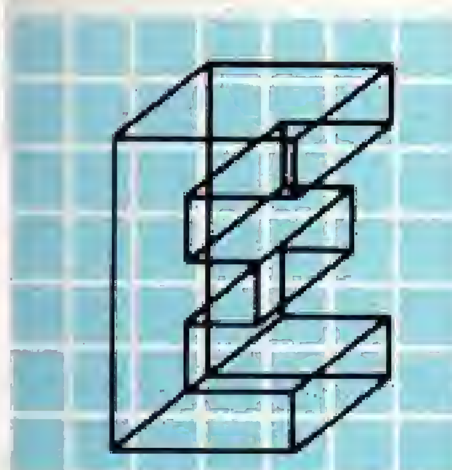
Spectrum

- Cambiar la línea 130 por DIM E$(3,3).
- Recuerda que el SPECTRUM no tiene locate. Sustitúyelo por PRINT AT.
- Cambia el BEEP de la línea 1060 por BEEP 15,0.5.

IBM

- No hace falta realizar ningún cambio.

Simulación de la sentencia 'LOCATE' en el Commodore



L COMMODORE es uno de los primeros microordenadores que aparecieron en el mercado. Debido a esto, carece de muchos de los comandos y funciones que hoy en día son normales en todos los ordenadores. Tal es el caso de la sentencia 'LOCATE'. Esta sentencia nos permite colocar el cursor en cualquier lugar de la pantalla o, lo que es igual, nos permite escribir en cualquier parte de la pantalla.

Con la rutina que proponemos a continuación este problema se resuelve de una manera rápida y eficaz.

El COMMODORE permite al usuario un cierto control sobre el cursor, así como borrar la pantalla y colocar el cursor en la posición de 'HOME' (esquina superior izquierda).

Si en uno de nuestros programas queremos borrar la pantalla hacemos lo siguiente:

1. Escribir PRINT"
2. Pulsar la tecla SHIFT y sin soltarla pulsar la tecla HOME.
3. Cerrar comillas.

Haciendo todo esto nos aparecerá entre las dos comillas un cuadradito con un corazón dibujado. Este signo le dice al COMMODORE que borre la pantalla. Si pulsamos RETURN el COMMODORE lo hará así.

```
9950 REM ******************************************************
9951 REM *                                                    *
9952 REM *          RUTINA 'LOCATE' PARA COMMODORE            *
9953 REM *                                                    *
9954 REM ******************************************************
9955 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA *
9956 REM * ------------------------------------------------ *
9957 REM * X  = COLUMNA DONDE NOS QUEREMOS POSICIONAR        *
9958 REM * Y  = FILA DONDE NOS QUEREMOS POSICIONAR           *
9959 REM *                                                    *
9960 REM * LA RUTINA NO DEVUELVE VALORES                      *
9961 REM *                                                    *
9962 REM * VARIABLES QUE SE USAN INTERNAMENTE                 *
9963 REM * ----------------------------------                 *
9964 REM * CC = CONTADOR DE BUCLES                            *
9965 REM ******************************************************
9966 REM
9967 PRINT "<HOME>";
9968 FOR CC=1 TO X
9969    PRINT "<CURSOR DERECHA>";
9970 NEXT CC
9971 FOR CC=1 TO Y
9972    PRINT "<CURSOR ABAJO>";
9973 NEXT CC
9974 RETURN
```

Esta rutina simula perfectamente la sentencia 'LOCATE' y, para realizarla, nos basaremos en lo siguiente.

Si en vez de pulsar las teclas SHIFT y HOME hubiésemos pulsado las del cursor, entonces nos hubiese aparecido otro carácter que el COMMODORE interpretaría como: 'MUEVE EL CURSOR HACIA la dirección que hayamos indicado'.



Fig. 1.—Así aparecen en la pantalla del Commodore los caracteres que hacen que se mueva el cursor al ponerlos entre las comillas de un print.

El programa que proponemos como sustituto de la sentencia 'LOCATE' hace lo siguiente:

1. Coloca el cursor en la posición HOME.
2. Mueve el cursor tantas veces a la derecha como hayamos indicado (se mueve a la columna X).
3. Mueve el cursor tantas veces hacia abajo como hayamos indicado (se mueve a la fila Y).

```
 10 REM **********************************
 20 REM * PROGRAMA EJEMPLO PARA UTLIZAR *
 30 REM * LA RUTINA 1.1                  *
 40 REM **********************************
 50 REM
 60 PRINT "<SHIFT-HOME>"
 70 INPUT "En que columna me coloco ";X
 80 INPUT "Y en que fila            ";Y
 90 PRINT "<SHIFT-HOME>"
100 PRINT  "Me voy a colocar en la columna
    ";X;" de la fila ";Y
110 GOSUB 9950
120 PRINT "AQUI ESTOY"
130 END
```

4. Deja el cursor en dicha posición en espera de que imprimamos algo.

Sabiendo esto, ya podemos interpretar el listado. Por esto, a la hora de introducir el programa en nuestro ordenador, donde pone <HOME> tendremos que pulsar la tecla 'HOME', y donde pone <CURSOR DERECHA> pulsaremos la tecla que tiene un dibujo con una flecha hacia la derecha.

Para utilizar esta rutina, hay que decirle la columna a la que nos queremos mover poniendo su número en la variable X y la fila en la variable Y. Seguidamente haremos un GOSUB 9950.



Fig. 2.—Con esta rutina podremos simular el comando LOCATE en el Commodore.

Subrutina de entrada de datos de propósito general

En casi todos los programas que se realizan hoy en día es necesario que el usuario introduzca datos al ordenador: como su nombre, la fecha o cualquier otro dato que el ordenador desconozca y necesite para ejecutar el programa.

La rutina que damos a continuación es un procedimiento de entrada de datos de propósito general y que nos permitirá, al incluirla

en nuestros programas, pedirle al usuario cualquier dato que necesitemos para la ejecución del programa.

NOMBRE: ....
1.er APELLIDO: ....
2.º APELLIDO: ....
TELEFONO: ....
DIRECCION: .... N.º .....
PROVINCIA: ....
PAIS: ....

Fig. 3.—Casi todos los programas comerciales utilizan una rutina para introducir datos parecida a ésta.

Esta rutina está localizada a partir de la línea 9900 para que no se pueda confundir con el programa general. Con esto nuestros programas serán más modulares y estarán mejor estructurados.

La petición de datos podemos hacerla en cualquier parte de la pantalla. Con solo dar valores a las variables X e Y, éstas le dirán a la rutina en qué columna y en qué fila se colocará el cursor y empezará la recogida de datos.

El número máximo de caracteres que el usuario podrá introducir, para dar el dato que se le pide, vendrá dado por el programa, y se almacenará en la variable LO. Esta variable puede tomar cualquier valor siempre que no exceda de la longitud de una línea. La rutina detecta automáticamente si se ha llegado al número máximo de caracteres permitidos.

El usuario no tendrá por qué introducir, lógicamente, tantos caracteres como número máximo hayamos descrito.

También se puede restringir el tipo de caracteres que se pueden introducir. Así, por ejemplo, podemos hacer que sólo se acepten las mayúsculas, sólo las minúsculas, los números, todas las letras o todos los caracteres. Para realizar esta función se necesitan dos variables.

W$ - Nos dice el primer carácter permitido

M$ - Nos dice el último carácter permitido

Por ejemplo, si queremos que la rutina sólo acepte las letras mayúsculas desde la A a la Z pondremos en nuestro programa:

```
LET W$ = "A"
LET M$ = "Z"
```

La rutina está preparada para admitir en cualquier momento los carácter espacio en blanco ( ), punto (.) y guión (-).

Cuando la rutina retorne, bien porque el usuario pulse RETURN o bien porque agotó el número máximo de caracteres disponible para definir dicho dato, el resultado estará almacenado en la variable alfanumérica D$.

El funcionamiento del programa línea a línea es el siguiente:

Las líneas incluidas entre la 9900 y la 9921 son una breve descripción del funcionamiento del programa y de las variables usadas: internas y de uso propio de la rutina y externas o variables que se le pasan a la rutina. Este grupo de líneas lo podemos quitar, una vez introducido el programa y entendido su funcionamiento, para así ahorrar memoria.

La línea 9922 inicializa la variable alfanumérica D$ y la numérica LO. LO será el contador interno de los caracteres introducidos hasta un cierto momento por el usuario.

La línea 9923 coloca el cursor en la columna X de la fila Y. En el SPECTRUM hay que cambiar la sentencia LOCATE Y,X por PRINT AT Y, X. Para usar este programa en el COMMODORE, donde ponga LOCATE poner GOSUB 9950 e introducir la rutina para el COMMODORE que se ha dado anteriormente. En otros ordenadores, como el MSX, los argumentos de la sentencia LOCATE van al contrario, con lo que si vemos LOCATE Y,X, los dueños de dichos ordenadores tendrán que poner LOCATE X,Y.

Las líneas 9924, 9925 y 9926 imprimen en la pantalla tantos puntos (.) como caracteres le está permitido escribir al usuario. Esto se ha hecho de esta manera para que el usuario sepa hasta dónde puede escribir, pero se puede quitar o sustituir por cuadrados de colores, líneas, etc.

A partir de este momento se empieza la rutina de verdad. Se mira si se ha pulsado una tecla y se estudia el carácter tecleado para ver si está dentro de los límites permitidos.

La tecla DELETE sirve para borrar el carácter que está a la izquierda del cursor y mover el cursor un lugar hacia la izquierda. En la línea 9932 se pregunta si se ha pulsado esta tecla y alguna otra con anterioridad. En caso afirmativo, el contador de caracteres interno disminuye en uno, a la variable D$ se le quita el último carácter introducido (el que está más a la derecha. D$ = left$ (D$,LO)) y

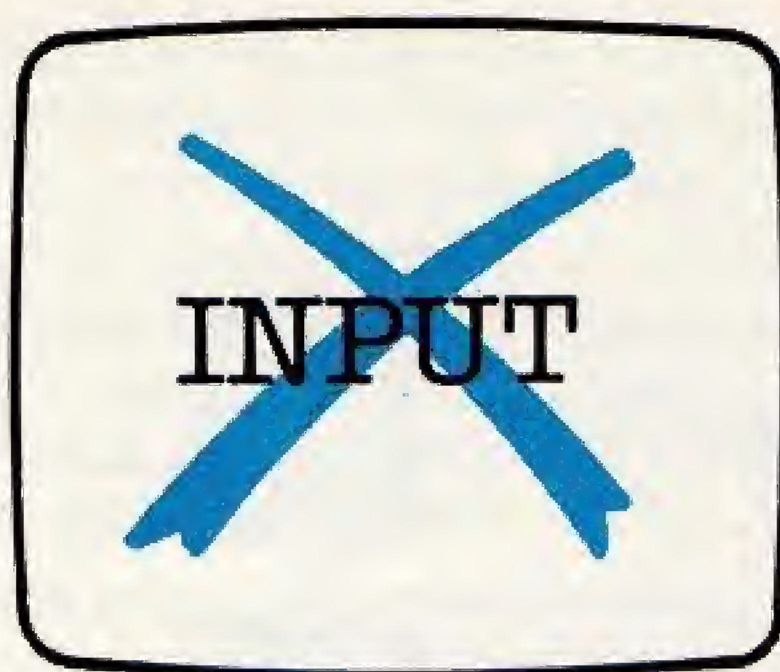


Fig. 4.—Con esta rutina ya te puedes olvidar de la incomodidad del INPUT.

ESTA RUTINA PERMITE LA ENTRADA DE DATOS DE UNA FORMA MAS ELEGANTE Y PRACTICA QUE SI SE HUBIESE HECHO CON UN INPUT

CUANDO NECESITES PEDIR UN NUMERO , M$ SERA IGUAL A "9" Y W$ A "0". COMO EL RESULTADO VENDRA EN D$, PARA RECUPERARLO EN FORMA NUMERICA PONDREMOS:

LET NU = VAL (D$)

se coloca el cursor un lugar más a la izquierda que el que ocupaba antes.

En la línea siguiente (la 9933) se pregunta si se ha pulsado espacio ( ), un punto (.) o un guión (-) para, en caso afirmativo, mandar el control del programa a la línea 9936 para imprimir lo pulsado.

Si se ha pulsado RETURN o ENTER o NEWLINE o INTRO (línea 9934) se hace GOTO a la línea 9942 para terminar y devolver el control al programa principal.

En la línea 9935 se comprueba si la tecla pulsada está dentro del rango de las permitidas y, si no lo está, se vuelve a la línea 9930.

A partir de este punto se imprime la tecla pulsada (9936), se suma uno a la posición en X del cursor (9937), se coloca el cursor en lugar hacia la derecha (9938), se concatena D$ con A$ para almacenar la tecla pulsada (9939) y se incrementa en uno el contador de caracteres introducidos (9940).

Si el número de caracteres introducidos es distinto del número máximo de caracteres permitidos se repite la operación para captar otro carácter (línea 9941). En caso contrario, se borran los puntos sobrantes a la derecha (9942) y se devuelve el control al programa principal.

Hay que advertir que como el SPECTRUM no tiene la función LEFT$, en la línea 9932, donde aparece esta función, habrá que poner: LET D$ = D$(TO LO).

```
9900 REM ****************************************************
9901 REM *       <<<< RUTINA DE ENTRADA DE DATOS >>>>       *
9902 REM *       VALIDA PARA MSX, IBM, AMSTRAD, SPECTRUM    *
9903 REM *       Y COMMODORE                                *
9904 REM ****************************************************
9905 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA *
9906 REM * ------------------------------------------------ *
9907 REM * X  = COLUMNA DONDE SE EMPIEZA A IMPRIMIR         *
9908 REM * Y  = FILA DONDE SE EMPIEZA A IMPRIMIR            *
9909 REM * LO = LONGITUD MAXIMA DEL DATO                    *
9910 REM * M$ = MAXIMO CARACTER PERMITIDO                   *
9911 REM * W$ = MINIMO CARACTER PERMITIDO                   *
9912 REM *                                                  *
9913 REM * EL DATO INTRODUCIDO SERA DEVUELTO EN D$          *
9914 REM *                                                  *
9915 REM * VARIABLES USADAS INTERNAMENTE                    *
9916 REM * ----------------------------                     *
9917 REM * LO = CONTADOR DE CARACTERES INTRODUCIDOS         *
9918 REM * CC = CONTADOR DE BUCLES                          *
9919 REM * A$ = RECOGE EL CARACTER TECLEADO                 *
9920 REM ****************************************************
9921 REM
9922 LET D$="" : LET LO=0
9923 LOCATE Y,X
9924 FOR CC=1 TO LO
9925    PRINT ".";
9926 NEXT CC
9927 LOCATE Y,X
```

```
9928 PRINT "_"
9929 LOCATE Y,X
9930 LET A$=INKEY$
9931 IF A$="" THEN GOTO 9930
9932 IF A$=CHR$(8) AND LO>O THEN LET LO=LO-1:LET D$=LEFT$(D$,LO) :LET X=X-1:LOCA
TE Y,X:PRINT "_.":LOCATE Y,X:GOTO 9930
9933 IF A$=" " OR A$="." OR A$="-" THEN GOTO 9936
9934 IF A$=CHR$(13) THEN GOTO 9942
9935 IF A$>M$ OR A$<W$ THEN GOTO 9930
9936 PRINT A$;"_"
9937 LET X=X+1
9938 LOCATE Y,X
9939 LET D$=D$+A$
9940 LET LO=LO+1
9941 IF LO<>LO THEN GOTO 9930
9942 LOCATE Y,X
9943 FOR CC=LO TO LO
9944    PRINT " ";
9945 NEXT CC
9946 RETURN
```

Como ejemplo veremos un programa que formaría parte de una agenda de nombres y teléfonos.

```
10 REM ************************************
20 REM * PROGRAMA DE DEMOSTRACION PARA    *
30 REM * VER EL FUNCIONAMIENTO DE LA RU-  *
40 REM * TINA DE ENTRADA DE DATOS.        *
50 REM *                                  *
60 REM * VALIDO PARA IBM, MSX, AMSTRAD,   *
70 REM * SPECTRUM Y COMMODORE             *
80 REM ************************************
90 REM
100 DIM R(4)
110 CLS:REM <-- EN EL COMMODORE SUSTITUIRLO POR: PRINT "<SHIFT-HOME>"
120 LOCATE 1,1:PRINT "AGENDA TELEFONICA"
130 PRINT "-----------------"
140 LOCATE 4,1:PRINT "NOMBRE: "
150 LOCATE 6,1:PRINT "APELLIDOS:"
160 LOCATE 8,1:PRINT "DIRECCION:"
170 LOCATE 10,1:PRINT "TELEFONO:"
180 FOR I=4 TO 8 STEP 2
190 LET Y=I
200 LET X=12
210 LET LO=20
220 LET M$="Z"
230 LET W$="A"
240 GOSUB 9900
250 LET R$((I-2)/2)=D$
260 NEXT I
270 LET Y=10
280 LET X=12
290 LET LO=8
300 LET M$="9"
310 LET W$="0"
320 GOSUB 9900
330 R$(4)=D$
340 END
```

Ejercicios resueltos

EJERCICIO N.º 1

¿Serías capaz de modificar la rutina de entrada de datos para que, en el caso de pedir un número, aceptase el principio y sólo al principio de la frase un guión? ¿Y que sólo admitiese un punto decimal?

```
9900 REM ******************************************************
9901 REM *      <<< RUTINA AMPLIADA ENTRADA DE DATOS >>>      *
9902 REM *                                                    *
9903 REM *     PARA MSX,IBM,AMSTRAD,COMMODORE Y SPECTRUM      *
9904 REM ******************************************************
9905 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA *
9906 REM * ------------------------------------------------- *
9907 REM * X  = COLUMNA DONDE SE EMPIEZA A IMPRIMIR          *
9908 REM * Y  = FILA DONDE SE EMPIEZA A IMPRIMIR             *
9909 REM * LO = LONGITUD MAXIMA DEL DATO                     *
9910 REM * M$ = MAXIMO CARACTER PERMITIDO                    *
9911 REM * W$ = MINIMO CARACTER PERMITIDO                    *
9912 REM *                                                    *
9913 REM * EL DATO INTRODUCIDO SERA DEVUELTO EN D$           *
9914 REM *                                                    *
9915 REM * VARIABLES USADAS INTERNAMENTE                     *
9916 REM * -----------------------------                    *
9917 REM * LO = CONTADOR DE CARACTERES INTRODUCIDOS          *
9918 REM * CC = CONTADOR DE BUCLES                           *
9919 REM * A$ = RECOGE EL CARACTER TECLEADO                  *
9920 REM ******************************************************
9921 REM
9922 LET D$="" : LET LO=0
9923 LET SW=0
9924 LOCATE Y,X
9925 FOR CC=1 TO LO
9926     PRINT ".";
9927 NEXT CC
9928 LOCATE Y,X
9929 PRINT "_"
9930 LOCATE Y,X
9931 LET A$=INKEY$
9932 IF A$="" THEN GOTO 9931
9933 IF A$=CHR$(8) AND LO>0 THEN LET LO=LO-1:LET SW=-(RIGHT$(D$,LO+1)="."):LET D
$=LEFT$(D$,LO) :LET X=X-1:LOCATE Y,X:PRINT "_.":LOCATE Y,X:GOTO 9931
9934 IF A$=" " THEN GOTO 9939
9935 IF A$="." AND SW=0 THEN LET SW=1:GOTO 9939
9936 IF A$="-" AND LO=0 THEN GOTO 9939
9937 IF A$=CHR$(13) THEN GOTO 9945
9938 IF A$>M$ OR A$<W$ THEN GOTO 9931
9939 PRINT A$;"_"
9940 LET X=X+1
9941 LOCATE Y,X
9942 LET D$=D$+A$
9943 LET LO=LO+1
9944 IF LO<>LO THEN GOTO 9931
9945 LOCATE Y,X
9946 FOR CC=LO TO LO
9947     PRINT " ";
9948 NEXT CC
9949 RETURN
```

SOLUCION

Las modificaciones que se pedían eran muy sencillas de realizar. Hay muchas maneras de hacerlo. Una solución posible es la que se ofrece en el programa 5.

La explicación de su funcionamiento os la dejamos a vosotros.

Impresión de mensajes de distintas formas

Los programas comerciales que se venden en las tiendas, aparte de ser más o menos prácticos o divertidos, suelen estar muy bien presentados para así hacer al usuario más agradable el programa. Uno de los trucos que más se utilizan es el de imprimir los mensajes de manera distinta a la usual. En este fascículo daremos una serie de rutinas que tú puedes incorporar a tus programas para hacerlos más vistosos.

Antes de empezar te recomiendo que introduzcas la rutina 6 y ejecutes el programa ejemplo (PROGRAMA 1.7) que te damos a continuación. Así te harás idea de las posibilidades de estas rutinas.

PRESENTACION
DE
MENSAJES

Fig. 5.—Es muy importante, en los programas, la forma en que aparecen los mensajes que se dan al usuario.

```
8900 REM ***********************************************************
8901 REM *                                                         *
8902 REM *     <<< IMPRESION HORIZONTAL TIPO TELETIPO >>>          *
8903 REM *                                                         *
8904 REM ***********************************************************
8905 REM *                                                         *
8906 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA               *
8907 REM * -----------------------------------------               *
8908 REM *                                                         *
8909 REM * X  = COLUMNA DONDE SE EMPEZARA A IMPRIMIR EL MENSAJE    *
8910 REM * Y  = FILA DONDE SE EMPEZARA A IMPRIMIR EL MENSAJE       *
8911 REM *                                                         *
8912 REM * M$ = MENSAJE A IMPRIMIR                                 *
8913 REM *                                                         *
8914 REM * LF = ANCHO DE LA PANTALLA O VENTANA EN CARACTERES       *
8915 REM * TI = RETARDO ENTRE LETRA Y LETRA                        *
8916 REM *                                                         *
8917 REM * LA RUTINA NO DEVUELVE NINGUN VALOR                      *
8918 REM *                                                         *
8919 REM * VARIABLES UTILIZADAS INTERNAMENTE POR LA RUTINA         *
8920 REM * -----------------------------------------------         *
8921 REM *                                                         *
8922 REM * CC = CONTADOR DE BUCLE                                  *
8923 REM * CD = CONTADOR DE BUCLE                                  *
8924 REM *                                                         *
8925 REM ***********************************************************
8926 REM
8927 LET X=INT((LF-LEN(M$))/2)
8928 LOCATE Y,X
8929 FOR CC=1 TO LEN(M$)
8930     PRINT MID$(M$,CC,1);
8931     PLAY "L64 D"
8932     FOR CD=1 TO TI
8933     NEXT CD
8934 NEXT CC
8935 RETURN
```

Como puedes apreciar, esta rutina imprime el mensaje letra a letra haciendo un sonido cada vez que imprime una. La velocidad de impresión se puede regular dando valores más o menos grandes a la variable TI.

```
PROGRAMA 1.7
============

 10 REM ***************************
 20 REM * PROGRAMA EJEMPLO PARA   *
 30 REM * UTILIZAR LA RUTINA 1.6 *
 40 REM ***************************
 50 REM
 60 LET Y=10
 70 LET M$="ESTO ES UN EJEMPLO DE COMO"
 80 LET LF=32
 90 LET TI=50
100 GOSUB 8900
110 LET Y=11
120 LET M$="FUNCIONA ESTA RUTINA."
130 GOSUB 8900
140 END
```

Las variables que tenemos que pasar son las siguientes:

Y = Le dice a la rutina en qué columna va a imprimir.

M$ = Es el mensaje a imprimir.

LF = Es la anchura de la pantalla en caracteres. Esta variable puede tomar cualquier valor entre 1 y el ancho de la pantalla. Si ponemos un valor menor que el de la pantalla, como en la línea 8927 se ajusta el texto para centrarlo, el mensaje aparecerá desplazado hacia la izquierda.

TI = Es el retardo entre letra y letra.

Su funcionamiento es muy sencillo. En la línea 9927 se centra el texto. Si no queremos que nos lo centre, tendremos que darle a la rutina la variable X y hacer GOSUB 8928 en vez de GOSUB 8900. La variable X indica la columna donde empezará la impresión.

En la línea 8928 colocamos el cursor donde corresponde.

A partir de la línea siguiente (8929) comienza un bucle, gracias al cual vamos imprimiendo la cadena letra a letra usando la función MID$. Los usuarios del SPECTRUM tienen que sustituir las líneas 8928, 8930 y 8931 por:

```
8928 PRINT AT, Y,X;
8930 PRINT M$ (CC);
8931 BEEP 15,0,2
```

A continuación viene un bucle en vacío (sin nada dentro del bucle) que lo que hace es perder tiempo antes de imprimir la siguiente letra.

Otra forma de imprimir mensajes es la que realiza la rutina 8 que aparece a continuación.

```
8800 REM ************************************************************
8801 REM *   <<< IMPRESION HORIZONTAL CON DESPLAZAMIENTO >>>       *
8802 REM *                                                          *
8803 REM *   PARA MSX,IBM,AMSTRAD,COMMODORE Y SPECTRUM              *
8804 REM ************************************************************
8805 REM *                                                          *
8806 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA                *
8807 REM * ----------------------------------------                 *
8808 REM *                                                          *
8809 REM * X  = COLUMNA DONDE EMPIEZA LA IMPRESION                  *
8810 REM * Y  = FILA DONDE COMIENZA LA IMPRESION                    *
8811 REM *                                                          *
8812 REM * M$ = MENSAJE A IMPRIMIR                                  *
8813 REM *                                                          *
8814 REM * LF = ANCHURA MAXIMA DEL MENSAJE                          *
8815 REM * TI = PAUSA ENTRE LETRAS                                  *
8816 REM *                                                          *
8817 REM * LA RUTINA NO DEVUELVE VALORES                            *
8818 REM *                                                          *
8819 REM * VARIABLES USADAS INTERNAMENTE POR LA RUTINA              *
8820 REM * ------------------------------------------               *
8821 REM *                                                          *
8822 REM * CC = CONTADOR DE BUCLE                                   *
8823 REM * CD = CONTADOR DE BUCLE                                   *
8824 REM *                                                          *
8825 REM * B$ = STRING DE BLANCOS AUXILIAR                          *
8826 REM *                                                          *
8827 REM ************************************************************
8828 REM
8829 LET B$=""
8830 FOR CC=1 TO LF+2
8831     LET B$=B$+" "
8832 NEXT CC
8833 FOR CC=1 TO LEN(B$+M$)+1
```

```
8834     LOCATE Y,X
8835     PRINT MID$(B$+M$+B$,CC,LF)
8836     PLAY "L64 D"
8837     FOR CD=1 TO TI
8838     NEXT CD
8839 NEXT CC
8840 RETURN
```

Esta rutina hace que el mensaje vaya apareciendo por la derecha de la pantalla y se desplace hacia la izquierda carácter a carácter.

Las variables que hay que pasarle son las mismas que las de la rutina anterior aunque funciona de forma distinta.

Para ver un ejemplo de su funcionamiento utiliza el programa 7 intercalando la siguiente línea:

65 LET X = 1

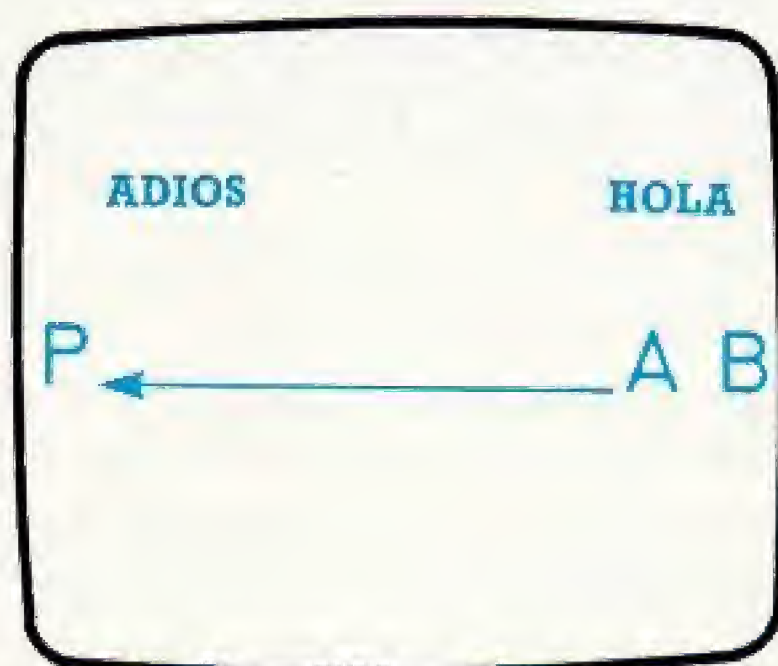


Fig. 6.—El programa 1.8 hace que las letras entren por la derecha de la pantalla y desaparezcan por la izquierda.

El programa, aunque es algo más complicado que el anterior, es muy sencillo.

Lo que hace esta subrutina es crear una variable alfanumérica de longitud LF + 2 llena de caracteres en blanco (B$). Esta se unirá, a la hora de imprimir, con la variable que lleva el mensaje (M$). La rutina imprime series de LF caracteres. La primera vez que se realiza el bucle, que empieza en la línea 8833, se imprimen los primeros LF caracteres de la variable B$. En la segunda vuelta se imprimen los caracteres que van desde el segundo a LF + 1. La tercera desde el carácter tres a LF + 2. La cuarta desde el cuarto a LF + 3. En LF + 3 empieza a aparecer la variable M$ en la pantalla (sólo el primer carácter).

Al seguir ejecutándose este bucle dará la impresión de que el mensaje entra por la derecha de la pantalla y se desplaza a la izquierda.

En el SPECTRUM habría que cambiar las líneas 8834, 8835 y 8836 por:

8834 PRINT AT Y,X;
8835 PRINT B$ + M$ + B$ (CC TO CC + LF)
8836 BEEP 15,0.1

El siguiente programa, el 9, imprime de una manera muy distinta a lo que hemos visto hasta ahora. En este caso las letras van a salir disparadas desde una línea y van a llegar a otra donde se pararán e irán formando el mensaje. La mejor forma de ver esto es introducir la rutina 9 y ejecutarla con el programa 10.

```
8600 REM ****************************************************
8601 REM *  <<< IMPRESION HORIZONTAL DE ABAJO ARRIBA >>>    *
8602 REM *                                                  *
8603 REM *  PARA MSX,IBM,AMSTRAD,COMMODORE Y SPECTRUM       *
8604 REM ****************************************************
8605 REM *                                                  *
8606 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA        *
8607 REM * -----------------------------------------        *
8608 REM * Y1 = LINEA DE LA QUE PARTE EL EMNSAJE            *
8609 REM * Y2 = LINEA A LA QUE LLEGA EL MENSAJE             *
8610 REM * X  = COLUMNA DONDE EMPIEZA EL MENSAJE            *
8611 REM * TI = RETARDO ENTRE LETRA Y LETRA                 *
8612 REM * M$ = MENSAJE A IMPRIMIR                          *
8613 REM *                                                  *
8614 REM * LA RUTINA NO DEVUELVE NINGUN VALOR               *
8615 REM *                                                  *
8616 REM * VARIABLES INTERNAS DE LA RUTINA                  *
8617 REM * -------------------------------                  *
8618 REM * CC = CONTADOR DE BUCLE                           *
8619 REM * CD = CONTADOR DE BUCLE                           *
8620 REM * CE = CONTADOR DE BUCLE                           *
8621 REM * ST = DIRECCION DEL MOVIMIENTO                    *
8622 REM *                                                  *
8623 REM ****************************************************
```

```
8624 REM
8625 FOR CC=1 TO LEN(M$)
8626 IF MID$(M$,CC,1)=" " THEN GOTO 8638
8627 LET ST=1
8628 IF Y1>Y2 THEN LET ST=-1
8629    FOR CD=Y1+ST TO Y2 STEP ST
8630        LOCATE CD,X-1+CC
8631        PRINT MID$(M$,CC,1)
8632        LOCATE CD-ST,X-1+CC
8633        PRINT " "
8634        FOR CE=1 TO TI
8635        NEXT CE
8636    NEXT CD
8637 PLAY "L64 A"
8638 NEXT CC
8639 RETURN
```

```
10 REM ****************************
20 REM * PROGRAMA EJEMPLO PARA LA *
30 REM * RUTINA 1.9               *
40 REM ****************************
50 REM
60 LET Y1=20
70 LET Y2=5
80 LET X=1
90 LET TI=20
100 LET M$="PARECE QUE SON BALAS."
110 GOSUB 8600
```

Como puedes apreciar, la ejecución de la rutina es bastante espectacular.



Fig. 7.—Con el programa 9 las letras salen disparadas, como balas, para formar una frase.

Las variables que hay que pasarle a la rutina son las siguientes:

Y1 = Línea de la que parten las letras.
Y2 = Línea a la que llegan las letras.

Estas dos variables tienen que tener distinto valor, pero cualquiera de ellas puede ser mayor que la otra. Por ejemplo, si decimos que Y1 = 20 y Y2 = 5, entonces las letras irán de abajo hacia arriba; sin embargo, si decimos que Y1 = 5 y Y2 = 20, las letras irán de arriba a abajo.

X = columna donde empieza la impresión.
TI = retardo entre letra y letra.
M$ = mensaje de imprimir.

Su funcionamiento es el siguiente:

Se hace un bucle desde la primera a la última letra del mensaje (línea 8625). Se analiza si el movimiento es hacia arriba (ST = -1) o hacia abajo (ST = 1). A partir de este momento (línea 8630) se empieza a imprimir cada letra en todas las filas que van desde Y1 hasta Y2. De esta forma se consigue una sensación de movimiento. Cuando llegan a Y2 se quedan paradas.

Cada vez que una letra llega a su lugar se espera un tiempo mediante un bucle vacío (líneas (8634, 8635)), se emite un ruidito (8627) y se continúa con la siguiente letra.

Para el SPECTRUM hay que cambiar las líneas 8626, 8630, 8631, 8632 y 8637 por las siguientes:

```
8626 IF M$ (CC) = " " THEN GOTO 8638
8630 PRINT AT CD, X-1 + CC
8631 PRINT M$ (CC)
8632 AT CD - ST, X - 1 + CC;
8637 BEEP 15,0.1
```

A continuación, ya para terminar, tenemos una rutina muy parecida a la anterior pero con mensajes que se disparan horizontalmente en vez de verticalmente. Con el programa 11 podremos imprimir mensajes de derecha a izquierda. Cada letra entrará por la pantalla por la derecha, se moverá muy rápido hacia la izquierda y se colocará en el lugar del mensaje que le corresponda.

```
8500 REM ***************************************************************
8501 REM *   <<< IMPRESION HORIZONTAL DE DERECHA A IZQUIERDA >>>   *
8502 REM *   PARA IBM,MSX,AMSTRAD,COMMODORE Y SPECTRUM              *
8503 REM *                                                          *
8504 REM ***************************************************************
```

```
8505 REM *                                                        *
8506 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA              *
8507 REM * -----------------------------------------              *
8508 REM * X1 = COLUMNA DE LA QUE PARTE EL MENSAJE                *
8509 REM * X2 = COLUMNA A LA QUE LLEGA EL MENSAJE                 *
8510 REM * Y  = FILA EN LA QUE SE IMPRIME EL MENSAJE              *
8511 REM * TI = RETARDO ENTRE LETRA Y LETRA                       *
8512 REM * M$ = MENSAJE A IMPRIMIR                                *
8513 REM *                                                        *
8514 REM * LA RUTINA NO DEVUELVE NINGUN VALOR                     *
8515 REM *                                                        *
8516 REM * VARIABLES UTILIZADAS POR LA RUTINA                     *
8517 REM * ----------------------------------                     *
8518 REM * CC = CONTADOR DE BUCLE                                 *
8519 REM * CD = CONTADOR DE BUCLE                                 *
8520 REM * CE = CONTADOR DE BUCLE                                 *
8521 REM *                                                        *
8522 REM **********************************************************
8523 REM
8524 LET X2=X2-1
8525 FOR CC=1 TO LEN(M$)
8526 IF MID$(M$,CC,1)=" " THEN GOTO 8535
8527    LET X2=X2+1
8528    FOR CD=X1 TO X2 STEP -1
8529       LOCATE Y,CD
8530       PRINT MID$(M$,CC,1);" "
8531       FOR CE=1 TO TI
8532       NEXT CE
8533    NEXT CD
8534    PLAY "L64 A"
8535 NEXT CC
```

Las variables utilizadas son las siguientes:

X1 = Columna de la que parte cada letra.

X2 = Columna a la que llega la primera letra. La segunda, como se coloca a la derecha de la primera, se pondrá en la columna X2 + 1, la tercera en X 2 + 2 y así sucesivamente.

Y = Fila en la que se desarrolla el mensaje.

TI = Tiempo de retardo entre letra y letra.

M$ = Mensaje a imprimir.

En el SPECTRUM habría que cambiar las líneas 8526, 8529, 8530 y 8534 por:

```
8526 IF M$ (CC) = " " THEN GOTO 8535
8529 PRINT AT Y, CD;
8530 PRINT M$ (CC); " "
8534 BEEP 15,0.1
```

NOTA

EN MUCHOS PROGRAMAS DE ESTE FASCICULO APARECE LA INSTRUCCION:

PLAY

ESTA ES VALIDA PARA IBM y MSX

Para el Spectrum cambiarla por: BEEP 15,0.1
Para el Amstrad cambiarla por: BEEP
Para el Commodore quitarla

Ejercicios resueltos

EJERCICIO N.º 2

¿Serías capaz de hacer un programa como el 11, pero que, en vez de escribir de derecha a izquierda, escribiese de izquierda a derecha, dejando el mensaje en su orden correcto? ¿Y en orden inverso?

SOLUCION

Como siempre, a un mismo problema se le pueden dar múltiples soluciones. En este caso la solución es casi inmediata, pues el programa que nos ha quedado es casi igual que el 11. El programa solución es el 12.

Para que el programa escribiese a la inversa sólo hay que cambiar la línea 8525 por:

8525 FOR CC = 1 TO LEN (M$)

y el mensaje saldrá escrito al contrario.

EJERCICIO N.º 3

¿Cómo modificarías el programa 11 para que fuese como el 9 y dejase, en un solo programa, escribir de derecha a izquierda y de izquierda a derecha?

SOLUCION

En este caso el problema es algo más difícil, pero no imposible. Una de las solucio-

```
8500 REM *************************************************************
8501 REM *    <<< IMPRESION HORIZONTAL DE IZQUIERDA A DERECHA >>>     *
8502 REM *                                                            *
8503 REM *    PARA IBM,MSX,AMSTRAD,COMMODORE Y SPECTRUM               *
8504 REM *************************************************************
8505 REM *                                                            *
8506 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA                  *
8507 REM * -----------------------------------------                  *
8508 REM * X1 = COLUMNA A LA QUE LLEGA EL MENSAJE                     *
8509 REM * X2 = COLUMNA DE LA QUE PARTE EL MENSAJE                    *
8510 REM * Y  = FILA EN LA QUE SE IMPRIME EL MENSAJE                  *
8511 REM * TI = RETARDO ENTRE LETRA Y LETRA                           *
8512 REM * M$ = MENSAJE A IMPRIMIR                                    *
8513 REM *                                                            *
8514 REM * LA RUTINA NO DEVUELVE NINGUN VALOR                         *
8515 REM *                                                            *
8516 REM * VARIABLES UTILIZADAS POR LA RUTINA                         *
8517 REM * ----------------------------------
8518 REM * CC = CONTADOR DE BUCLES                                    *
8519 REM * CD = CONTADOR DE BUCLES                                    *
8520 REM * CE = CONTADOR DE BUCLES                                    *
8521 REM *                                                            *
8522 REM *************************************************************
8523 REM
8524 LET X2=X2+1
8525 FOR CC=LEN(M$) TO 1 STEP -1
8526    LET X2=X2-1
8527    IF MID$(M$,CC,1)=" " THEN GOTO 8535
8528    FOR CD=X1 TO X2
8529       LOCATE Y,CD
8530       PRINT " ";MID$(M$,CC,1)
8531       FOR CE=1 TO TI
8532       NEXT CE
8533    NEXT CD
8534    PLAY "L64 A"
8535 NEXT CC
```

nes que se nos puede ocurrir la encontrarás en el programa 13.

Como siempre en este apartado, no explicamos los ejercicios para que tú mismo seas capaz de comprenderlo. Si lo consigues, lo cual no es nada difícil, habrás comprendido mucho más que si te lo explicamos.

```
8500 REM *************************************************************
8501 REM *    <<< IMPRESION HORIZONTAL DE IZQUIERDA A DERECHA >>>     *
8502 REM *    <<<              Y DE DERECHA A IZQUIERDA        >>>    *
8503 REM *                                                            *
8504 REM *    PARA IBM,MSX,AMSTRAD,COMMODORE Y SPECTRUM               *
8505 REM *************************************************************
8506 REM *                                                            *
8507 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA                  *
8508 REM * -----------------------------------------                  *
8509 REM * X1 = COLUMNA A LA QUE LLEGA EL MENSAJE                     *
8510 REM * X2 = COLUMNA DE LA QUE PARTE EL MENSAJE                    *
8511 REM * Y  = FILA EN LA QUE SE IMPRIME EL MENSAJE                  *
8512 REM * TI = RETARDO ENTRE LETRA Y LETRA                           *
8513 REM * M$ = MENSAJE A IMPRIMIR                                    *
8514 REM *                                                            *
8515 REM * LA RUTINA NO DEVUELVE NINGUN VALOR                         *
8516 REM *                                                            *
8517 REM * VARIABLES UTILIZADAS POR LA RUTINA                         *
8518 REM * ----------------------------------                         *
8519 REM * CC = CONTADOR DE BUCLES                                    *
8520 REM * CD = CONTADOR DE BUCLES                                    *
8521 REM * CE = CONTADOR DE BUCLES                                    *
8522 REM *                                                            *
8523 REM *************************************************************
8524 REM
8525 LET L2=1
8526 LET L1=LEN(M$)
8527 LET ST=1
8528 IF X1>X2 THEN LET ST=-1 : LET L2=L1: LET L1=1
8529 LET X2=X2+ST
8530 FOR CC=L1 TO L2 STEP -ST
8531    LET X2=X2-ST
8532    IF MID$(M$,CC,1)=" " THEN GOTO 8542
8533    FOR CD=X1 TO X2 STEP ST
8534       LOCATE Y,CD
8535       IF ST=1 THEN PRINT " ";
8536       PRINT MID$(M$,CC,1);
8537       IF ST=-1 THEN PRINT " "
8538       FOR CE=1 TO TI
8539       NEXT CE
8540    NEXT CD
8541    PLAY "L64 A"
8542 NEXT CC
```

NOTA PARA LOS USUARIOS DEL SPECTRUM

Las variables índices de los bucles, en el SPECTRUM, sólo pueden tener un carácter como nombre. Por eso, si vemos un programa que pone:

FOR CC = 1 TO 30

este nombre (CC) no nos será aceptado.

Para remediar esto se propone coger la segunda letra del nombre de la variable. Por ejemplo, si aparece:

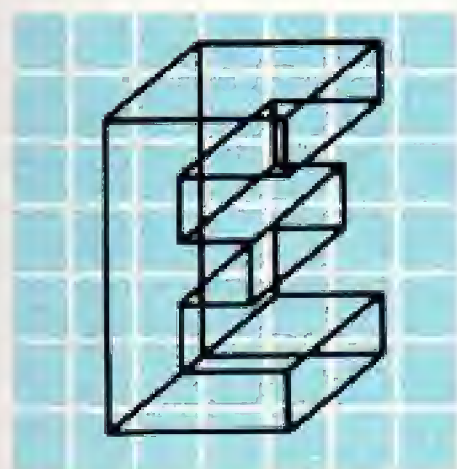
FOR CE = 1 TO TI

nosotros pondremos:

FOR E = 1 TO TI

Hay que tener cuidado con que ésta variable (E) no exista ya en el programa.

EL TALLER DEL HARDWARE



N esta sección vamos a tratar de presentar las posibilidades de los ordenadores personales, desde el punto de vista del hardware y en general como añadir alguna parte adicional, para un mayor disfrute de la máquina. Para que los circuitos que describamos puedan ser utilizados en los ordenadores más comunes, vamos a definir unas condiciones generales que deberán cumplir las conexiones y que puedan ser realizadas en cada uno de ellos. Si llevamos todas las señales necesarias a niveles apropiados y utilizamos un conector común, podremos conectar cualquiera de los dispositivos que diseñemos al ordenador que poseamos.

Los ordenadores sobre los que se mostrarán detalles son el ZX-Spectrum, Amstrad CPC-64, Comodore C-64 y el cada vez más difundido IBM-PC, por sus versiones originales o sus clones. Los ejemplos de aplicación son también utilizables en otras máquinas, pero necesitarán realizar alguna adaptación.

Ampliación de puertos de entrada/salida en un ordenador personal

La mayoría de los ordenadores personales poseen el conjunto de elementos necesarios para poder ejecutar programas de tipo recreativo, de proceso de textos, hoja electrónica y gráficos sin ningún equipo adicional. Pero hay muchas otras aplicaciones que pueden verse mejoradas añadiendo algún periférico sencillo. Sobre la idea de facilitar la conexión de nuevos dispositivos a nuestro ordenador vamos a empezar con la descripción de una tarjeta, que puede adaptarse a cualquiera de los ordenadores y que permite la ampliación del número de puertos de usuario. A partir de esta tarjeta se irán proponiendo nuevos circuitos que amplíen las posibilidades de nuestra máquina, con un mínimo de esfuerzo en cada proyecto.

El montar un conector universal en nuestro ordenador que nos permita añadirle los circuitos propuestos facilitará las conexiones futuras pero no es imprescindible, si sabemos convertir los circuitos descritos a las señales propias de nuestro equipo.

Vamos a empezar por describir los circuitos elementales que sirven para comunicar un dispositivo periférico con los buses de cada uno de los ordenadores más comunes: ZX-Spectrum, Amstrad CPC-64 y el IBM-PC. Se explican inicialmente los circuitos de interfaz más próximos a la CPU y en segundo lugar la conexión al conector común. Los bloques principales del conector universal son: Decodificador de dirección, puerto de salida y puerto de entrada. Casi todos los circuitos que vamos a describir necesitarán al menos un bloque de cada uno de éstos.

Decodificador de dirección

La unidad central activa las señales del bus del ordenador periódicamente de acuerdo con un reloj. En cada instrucción se producen pulsos que originan las transferencias de

datos entre las diferentes unidades. En las operaciones de comunicación con los periféricos, que son de entrada/salida, se activa el bus de direcciones con la dirección de la unidad que se accede, el acumulador o el registro utilizado en la transferencia y el bus de datos con la información. En el bus de control las diferentes señales indican el tipo de operación y el instante en el cual la transferencia se realiza. La lectura o escritura de la información por el circuito periférico debe hacerse de forma sincronizada con las señales del bus de control.

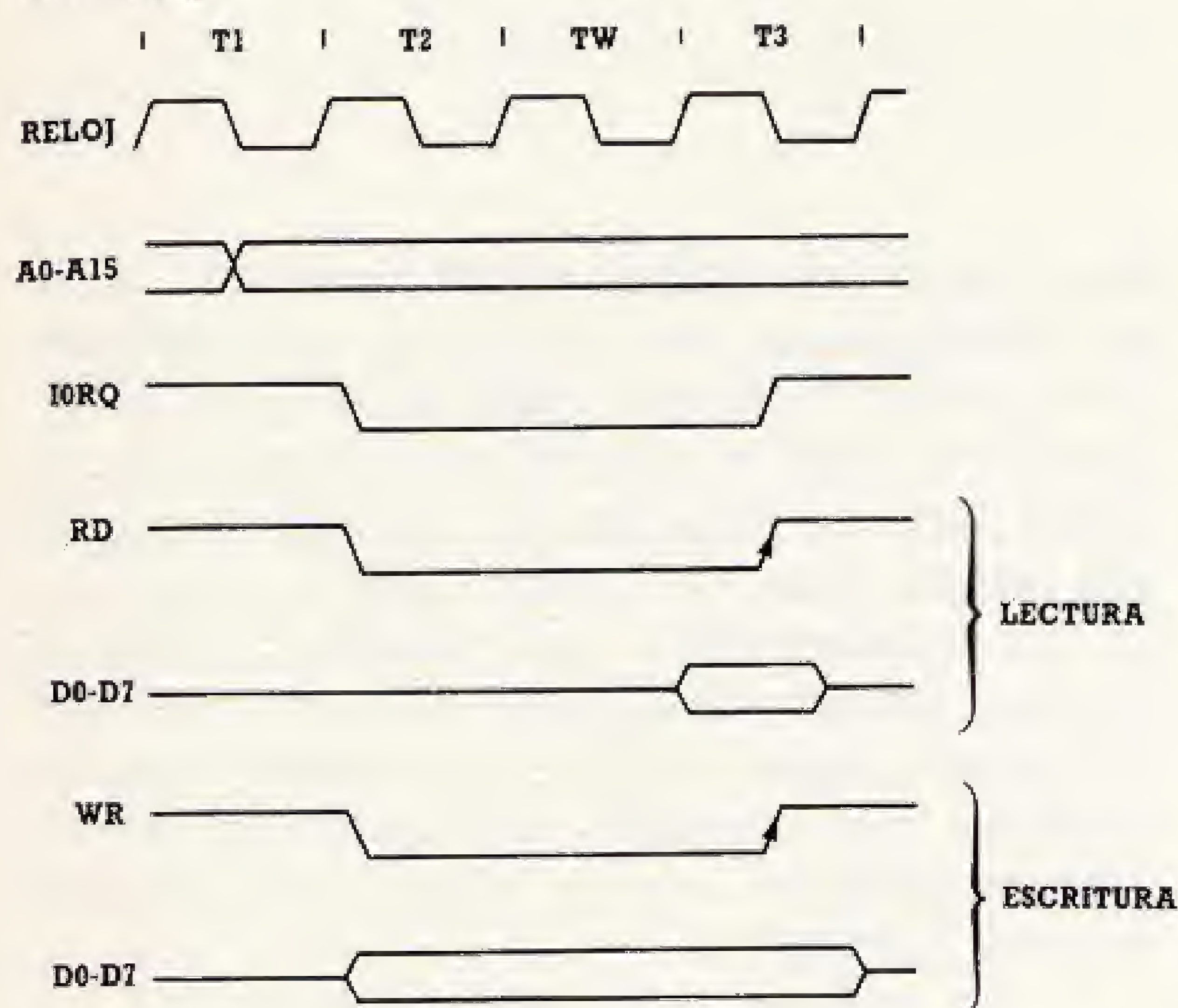


Fig. 1.—Señales generadas en operaciones E/S en Z80.

Para la comunicación con los periféricos es necesario en primer lugar asignar a cada conjunto de bits que tengan entidad común, una dirección que pueda ser accedida mediante la programación apropiada desde la unidad central. Es como su nombre para el ordenador. En segundo lugar será necesario diseñar el circuito que se ha de activar cuando se produzca la dirección bajo la cual ve la unidad central al periférico. La decodificación de una dirección consiste en la realización de la función AND con todos los bits que intervengan en la dirección, tomando la señal o su inversa según que en la dirección aparezca como 1 o como 0. El resultado de la función AND es 1 cuando todas las entradas son 1, por lo que para los bits de dirección que aparecen como 0, se tomará la señal complementaria. Con la señal obtenida se activará la carga de información desde el bus de datos al registro del periférico o se pasará desde el periférico al bus para que sea leída la información por la CPU.

Generalmente se necesita decodificar un conjunto consecutivo de direcciones por lo que sería necesario disponer de varias puertas AND con el correspondiente juego de inversores para cada señal que pueda necesitarse en modo invertido. Para estos casos se dispone de circuitos que incluyen la función de decodificador para 2, 3 o 4 bits, con lo que facilitan la obtención de señales de activación a partir de los bits de dirección de forma cómoda. Como añadido, estos circuitos permiten la selección mediante puentes de la dirección más apropiada para casos particulares, dentro del mapa de memoria o de puertos de entrada salida.

Veamos los primeros ejemplos de la forma de seleccionar dentro del espacio de direcciones de puerto, para cada uno de los ordenadores personales, con la posibilidad de dejar para el final la dirección definitiva. Se genera la señal -SEL. Esta señal deberá ser validada con las de lectura/escritura y entrada-salida, para realizar el acceso al periférico.

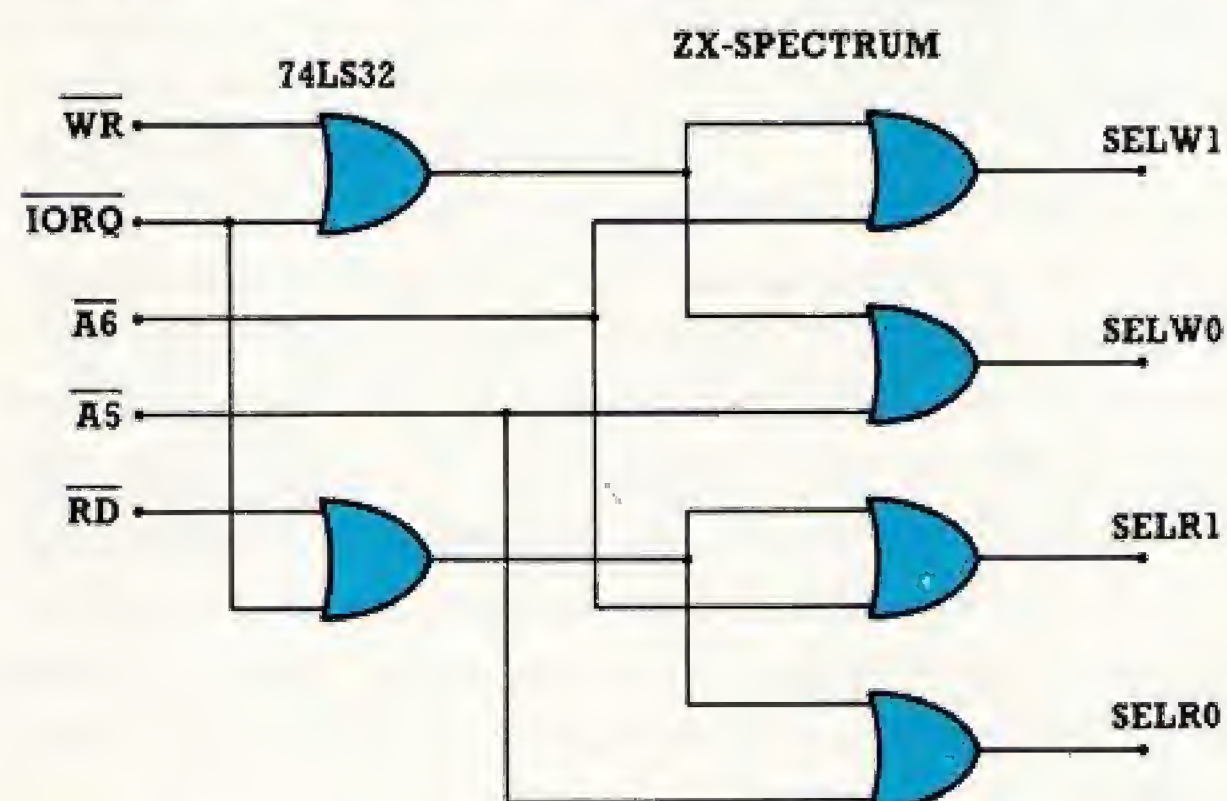


Fig. 2.—Circuito de decodificación para Spectrum.

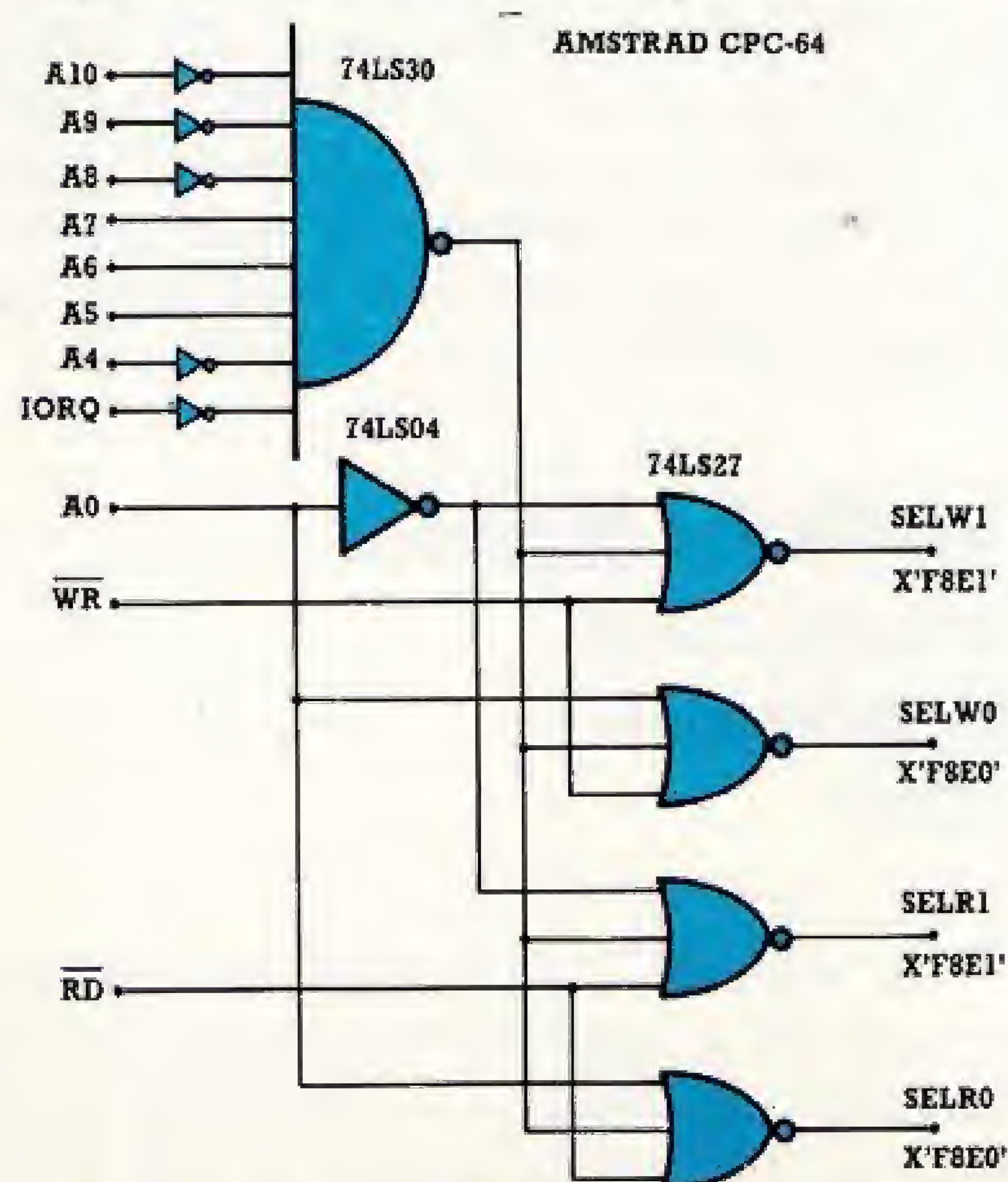


Fig. 3.—Circuito de decodificación para Amstrad CPC-64.

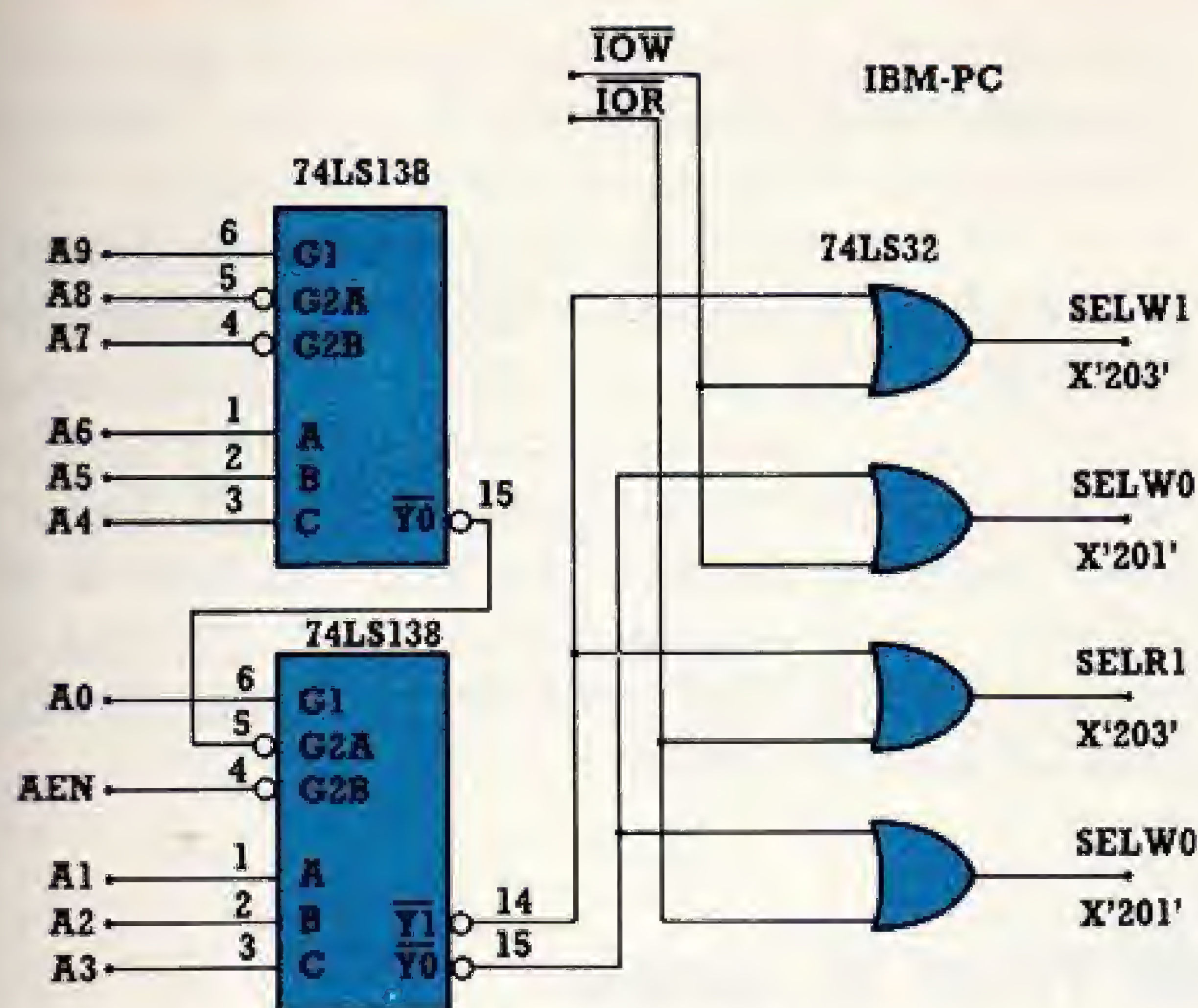


Fig. 4.—Circuito de decodificación IBM-PC.

En las señales de selección generadas, hemos asignado los nombres SELW0, SELW1, SELR0 y SELR1 a las direcciones correspondientes a zonas libres en cada una de las máquinas. En los ejemplos futuros se utilizarán estas denominaciones para referirse a los puertos de escritura (SELW) y de lectura (SELR).

Una forma todavía más cómoda de poder seleccionar la dirección de operación de un puerto es utilizar un comparador, una de cuyas entradas es programable desde una dirección fija de puerto. Así se permite desde el programa definir la dirección exacta donde se ubica el puerto de comunicación. Para la primera versión de la tarjeta no utilizaremos, sin embargo, esta forma de decodificación.

Las direcciones utilizadas en cada caso corresponden a zonas no utilizadas por otros dispositivos, y que por tanto no originan conflictos. Sin embargo, si ya se le han añadido otros periféricos adicionales, podría haber alguna incompatibilidad, que será necesario solventar mediante el cambio de alguno de los bits de dirección. El C-64 es el único de los micros descrito que utiliza direccionamiento de los dispositivos externos como si fueran memoria. Los demás poseen señales propias para las operaciones de entrada/salida. También presenta el C-64 la particularidad de poseer un conector para puerto de usuario. Es bidireccional pues utiliza un circuito interfaz con posibilidad de asignación del sentido de las transferencias para cada bit. Este puerto de usuario hace las misma funciones que los puertos que estamos describiendo, pero solamente para 8 bits.

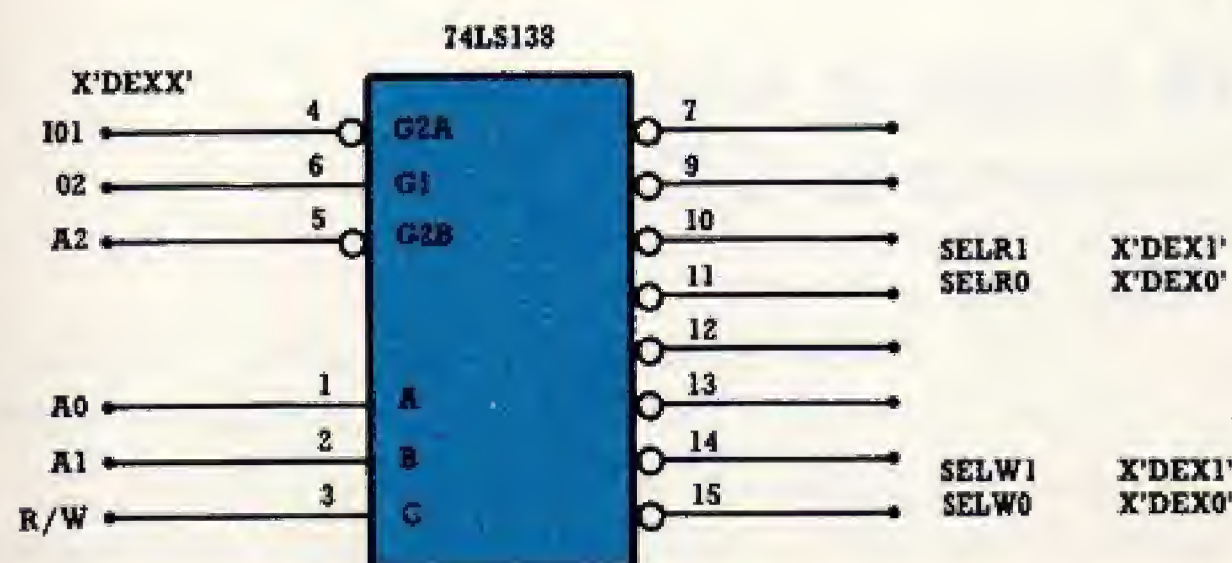


Fig. 5.—Circuito de decodificación para Commodore C-64.

La decodificación de direcciones puede hacerse de modo total o parcial. En el caso de que dispongamos de un equipo con pocos periféricos, puede ser cómodo y a la vez permite ahorrar algún circuito, dejar algún bit sin decodificar, con lo que se activará el circuito de decodificación tanto para la presencia de 1 como de 0 en la posición no decodificada. Esto hace que un mismo puerto pueda accederse mediante dos o más direcciones diferentes. Esto suele denominarse creación de "alias", pues la segunda dirección es un segundo "nombre" equivalente de la primera. Por cada bit dejado sin decodificar, duplicamos el espacio ocupado por cada puerto de entrada salida en el mapa de direcciones.

En general es recomendable hacer una decodificación completa, con un circuito parecido al que se ha mostrado en la figura para el IBM-PC, sin pretender apurar al máximo el número de circuitos. En las máquinas pequeñas es frecuente utilizar alguno de los bits altos de dirección para diferenciar cada zona de puertos, tal como se indica en la figura para el ZX-SPECTRUM. En esta máquina hay una asignación de bits del bus de direcciones para determinados periféricos.

Puerto de salida

El circuito más simple de salida hacia periférico está constituido por un registro de un solo bit, conectado a uno de los bit del bus de datos y que se se activa al producirse en el bus de direcciones la dirección correspondiente. El contenido del bus de datos se cargará en el registro de salida con el pulso generado por el decodificador y validado por las señales de escritura del bus de control. Si se utiliza registro de tipo D como el 74LS374, la transferencia se efectúa con el flanco de subida del pulso de control. Si se utiliza un registro de tipo transparente (74LS273), la carga desde el bus comienza a hacerse desde el mo-

mento en que la señal de control baja a cero. La información quedará retenida a partir de la subida de la señal de control.

Generalmente interesará poder disponer de los 8 bits, por lo que se conectará un circuito con posibilidad de almacenamiento de 8 bits. El circuito almacena la información transmitida desde el programa y la mantiene hasta que sea modificada o se apague la máquina. Si se necesita transmitir una secuencia de pulsos o una señal variable con el tiempo, será responsabilidad del programa el hacer aparecer en los tiempos oportunos las configuraciones de bits apropiados. Si es necesario modificar solamente uno de los bits del registro de 8 bits, será necesario dar a los restantes la misma información que poseían para que no aparezcan transiciones inoportunas en la salida. El circuito de registro indicado cumple estos requisitos.

El circuito integrado usado podría incluir alguna función adicional como por ejemplo contador o registro de desplazamiento, con lo que podrán realizarse operaciones directamente en la circuitería del adaptador, sin intervención de la unidad central, con el consiguiente ahorro de tiempo. La mayor parte de los circuitos de interfaz de los equipos actuales en OP, se diseñan utilizando circuitos integrados con posibilidad de programación, con lo que se obtiene mayor flexibilidad y ahorro si se selecciona el módulo apropiado. El circuito empleado en el esquema debe considerarse como básico. Además el circuito en general deberá tener en cuenta las condiciones iniciales, asegurando que no se producen efectos dañinos al encender el equipo.

La sentencia necesaria para escribir en el puerto de dirección DIR así montado la información contenida en la variable de un octeto A es la siguiente:

OUT DIR,A en BASIC para Spectrum, IBM-PC y Amstrad. Para Comodore como las operaciones con puertos son equivalentes a las de memoria, las instrucciones serán:,POKE DIR,A.

El SPECTRUM, sin embargo, utiliza los bits altos de direccionamiento a través de la ULA, para direccionar algunas líneas internas de la máquina, como por ejemplo la exploración del teclado se realiza sacando la dirección X'FE' en los bits bajos y de X'FE' a X'7F' los bits altos para cada una de las semifilas del teclado, con un bit a 0 para cada semifila. Estas direcciones de 16 bits utilizables desde BASIC son descompuestas por el programa de control realizando dos operaciones de entrada-salida de máquina, pero no son en realidad puertos con direccionamiento a 16 bits.

Con transferencias a través de puerto la velocidad máxima de salida de octetos está limitada por el número de ciclos necesarios para llevar la cuenta y sacar y meter los datos. Es del orden de 10.000 octetos por segundo el máximo posible con esta técnica. Si se necesita velocidad superior será necesario emplear DMA. Este método será descrito en otro capítulo posterior.

Puerto de entrada

El circuito más simple consiste en una puerta o un amplificador de bus, con tres estados, que se activen y den la señal en su salida al producirse el pulso de decodificación. Esta señal permite a la CPU leer el contenido del bus de datos en el momento en que el amplificador de entrada está activando el hilo correspondiente del bus.

En el esquema de ampliación de puertos de E/S se han utilizado circuitos de amplificación de tres estados de tipo 74LS244, con señal de control separada para cada 4 bits.

La sentencia para leer el puerto así montado es la siguiente:

A=INP(X) en BASIC para IBM-PC y A=IN X para Spectrum y Amstrad. Para C-64 la sentencia A=PEEK(DIR) realiza la misma función.

Muchos de los circuitos integrados que realizan funciones específicas poseen como circuito de salida amplificadores de tres estados, lo que les hace muy fáciles de conectar directamente al bus, controlando la señal de salida activa (OE) con la salida del decodificador de direcciones (-SEL) validada con IOR para el IBM-PC o con IOR y RD para los ordenadores que utilizan el micro Z80 como el SPECTRUM.

Tarjeta de ampliación de puertos de E/S

Con la descripciones de cómo realizar la decodificación de las direcciones y los puertos sencillos de entrada y salida en bloques separados, podemos diseñar para cada uno de los ordendores personales una tarjeta

de propósito general que nos permitirá la conexión de muy diferentes dispositivos que amplíen las posibilidades de nuestra máquina. El interés principal radica en que con esta sencilla tarjeta podremos realizar numerosos experimentos, añadiendo simplemente en cada caso los componentes específicos para el ensayo, pero teniendo ya la completa seguridad de que la tarjeta conectada al bus opera correctamente. Como sugerencia adicional

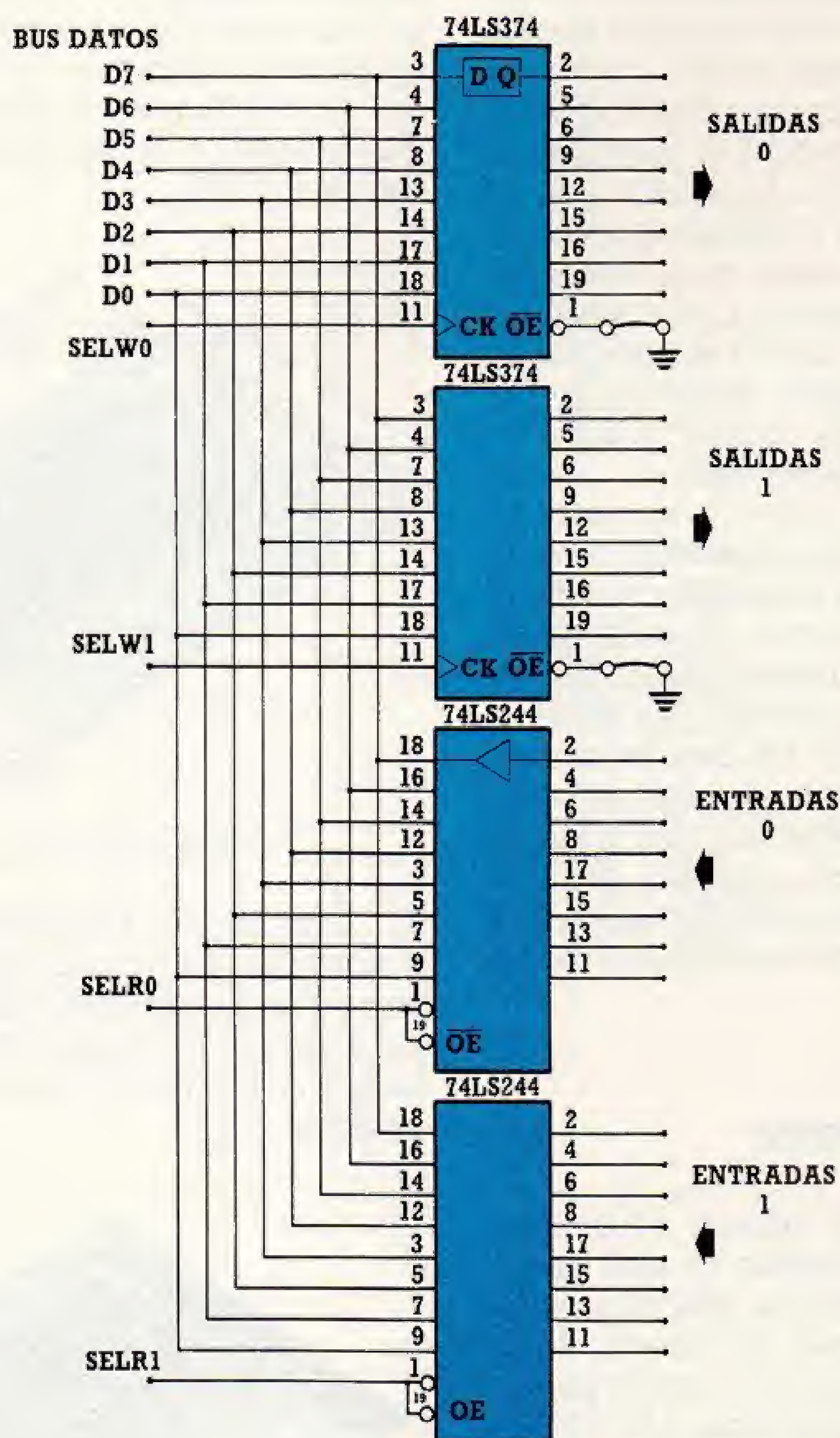


Fig. 6.—Tarjeta de ampliación de puertos de E/S.

MONTAJE PASO A PASO DE ESTA TARJETA

En próximos números se describirá el montaje paso a paso de las tarjetas explicadas en el texto (para cada uno de los ordenadores indicados) con gráficos explicativos y fotografías reales del montaje. Se incluirá, también, la tarjeta necesaria para los equipos MSX con esquemas y diagramas completos.

añadiremos que conviene que uno de los puertos de salida pueda ser cableado en paralelo con el de entrada de su misma dirección, para poder realizar interfaces bidireccionales en paralelo.

Para probar el funcionamiento de la tarjeta completa deberemos utilizar un polímetro para verificar que cada hilo de salida genera el nivel programado. Mejor todavía podremos montar un diodo LED en serie con una resistencia de 1 K. y conectado a la alimentación de +5 voltios. Conectado a las salidas, deberá lucir al programar un 0 y deberá apagarse al programar un 1. Igualmente para probar las entrada podremos conectar a +5 voltios o a masa cada uno de los hilos de entrada, mientras se ejecuta un pequeño programa en bucle que lea el puerto de entrada y presente de forma numérica el valor leído.

La tarjeta que contiene los circuitos de puerto la montaremos con conectores hacia el exterior, adaptándose a cada tipo de ordenador, para facilitar la conexión de los circuitos futuros. Este conector deberá incluir: Las señales de los puertos de entrada y de salida, la alimentación y masa. En los diseños tendremos presente la utilización de alguna de las señales de entrada y de salida para sincronización con el restro del periférico, es decir para las señales de indicación de cuando está disponible la información en cada uno de los sentidos.

Detalles constructivos

Las conexiones al exterior de las diferentes máquinas se presentan de muy diversas maneras. Los ordenadores Spectrum, Amstrad y Comodore disponen de conector único de expansión de tipo tarjeta de circuito impreso, macho, con patillas separadas 2,54 milímetros y en número de 28 x 2, 25 x 2 y 22 x 2 respectivamente. El IBM-PC dispone de varios conectores de expansión, 5-8 según el modelo, de tipo hembra de 31 x 2 contactos. En la figura se muestran los diferentes conectores con la denominación de cada contacto según la documentación original. Es importante resaltar que denominamos cara de arriba a la que normalmente contiene los componentes y cara de abajo la que presenta las soldaduras. Para facilitar la conexión de las tarjetas de los próximos proyectos y de los experimentos en general, se sugiere colocar en la tarjeta un conector hembra con los terminales de entrada-salida de los puertos. Para terminación de los proyectos definitivos hay otras soluciones igualmente operativas y económicas, como por ejemplo los conectores de tornillo prisionero (clemas o fichas), en la forma que indican los dibujos.

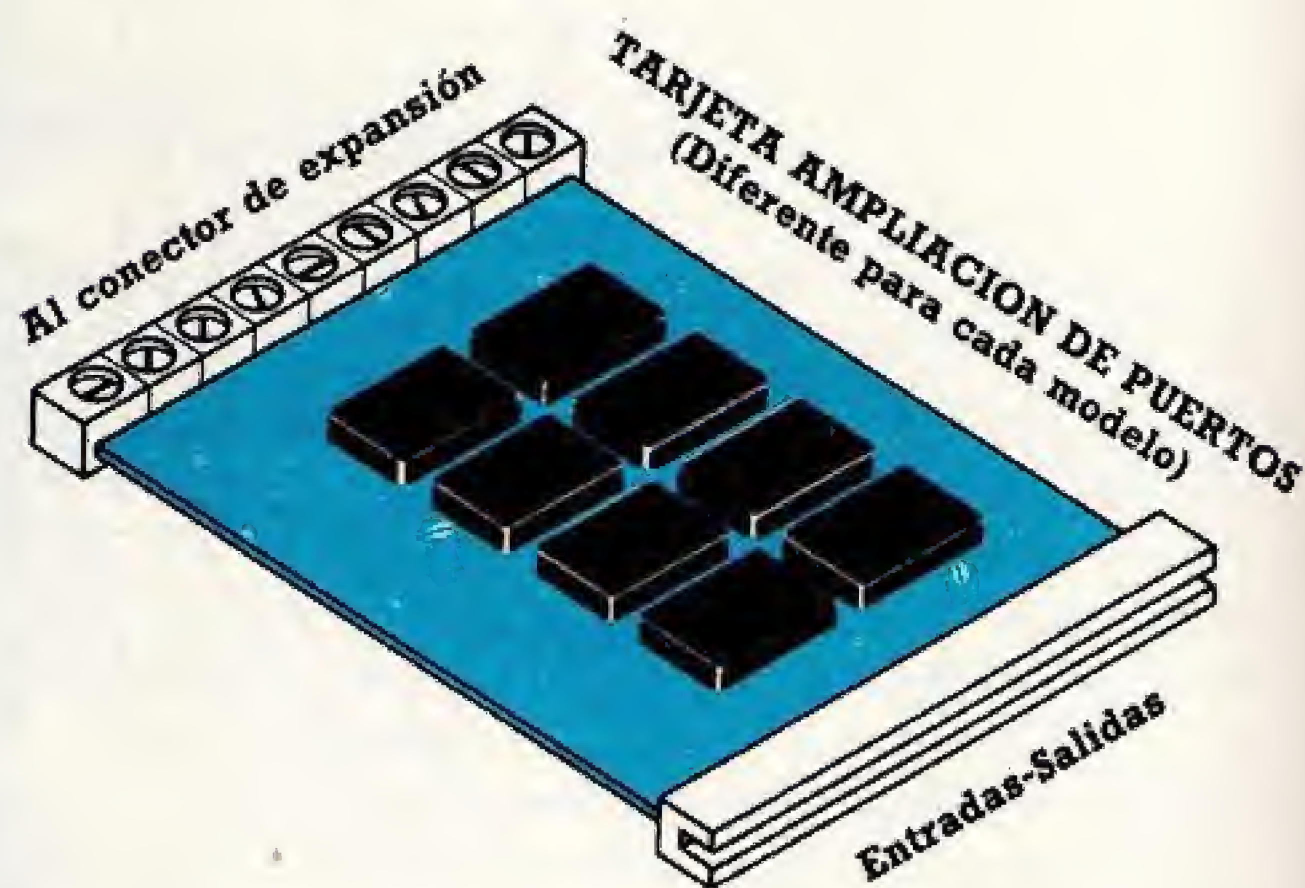


Fig. 7.—Ideas sobre el montaje de los proyectos.

Las tarjetas resultarán más económicas si se realizan en circuitos de tiras o circuito impreso de una sola cara, aunque requiere mayor espacio. Las conexiones entre circuitos se realizarán con puentes por el lado de los componentes.

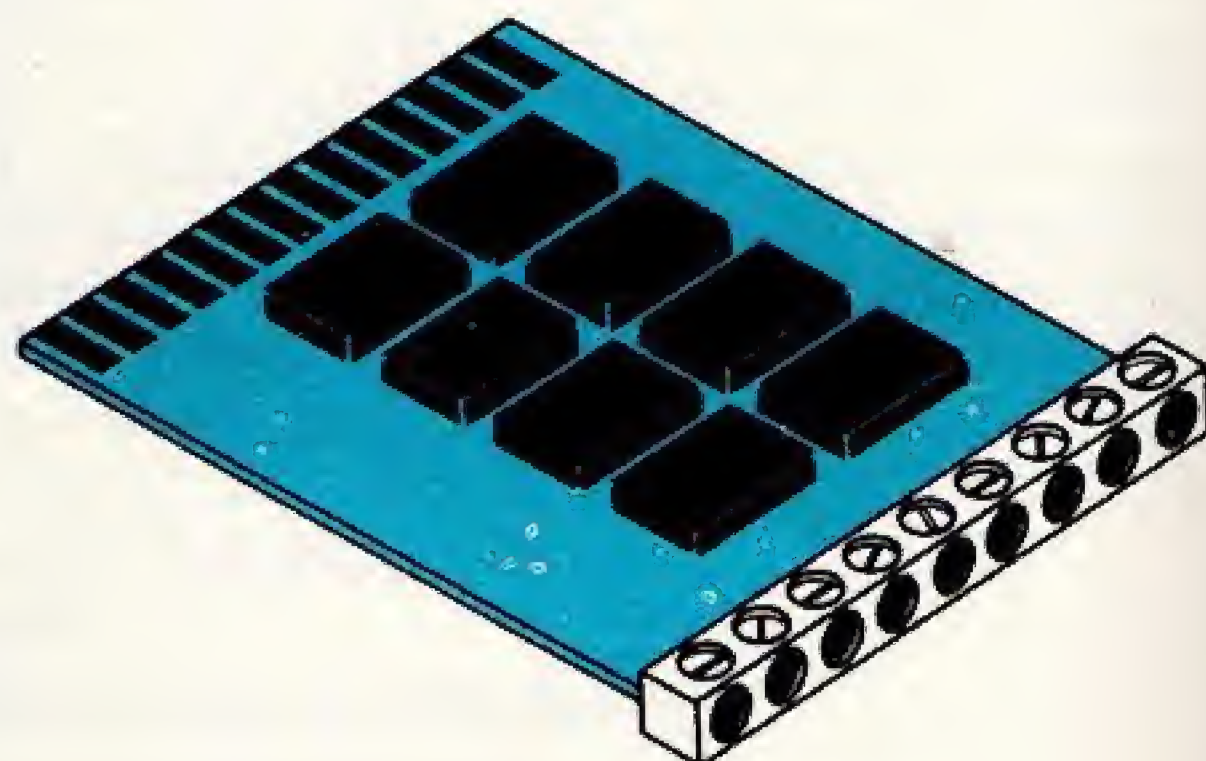


Fig. 8.—Tarjeta de proyectos.

Conectores de expansión

ZX-SPECTRUM

	B	A
1	A14	A15
2	A12	A13
3	+5V	D7
4	+9V	NC
5	⊏ RANURA	⊏
6	⊥	D0
7	⊥	D1
8	CLK	D2
9	A0	D6
10	A1	D5
11	A2	D3
12	A3	D4
13	-IORQGE	-INT
14	⊥	-NML
15	VIDEO	-HALT
16	Y	-MREQ
17	V	-IOREQ
18	⊔	-RD
19	-BUSREQ	-WR
20	-RESET	-5V
21	A7	-WAIT
22	A6	+12V
23	A5	+12V(nF)
24	A4	-M1
25	-ROMCS	-RFSH
26	-BUSACK	A8
27	A9	A10
28	A11	NC
	ABAJO SOLDADURA	ARRIBA COMPONENTES

IBM-PC

	B	A
1	⊥	-I10CWCK
2	RSTDRV	D0
3	+5V	D1
4	IRQ2	D2
5	-5V	D3
6	DRQ2	D4
7	-12V	D5
8		D6
9	+12V	D7
10	⊥	I/O CH RDY
11	-MEMW	AEW
12	-MEMR	A19
13	-IOW	A18
14	-IOR	A17
15	-DACK3	A16
16	DRQ3	A15
17	-DACK1	A14
18	DRQ1	A13
19	-DACK0	A12
20	CLK	A11
21	IRQ7	A10
22	IRQ6	A9
23	IRQ5	A8
24	IRQ4	A7
25	IRQ3	A6
26	-DACK2	A5
27	T/C	A4
28	ALE	A3
29	+5V	A2
30	OSC	A1
31	⊥	A0
	ABAJO SOLDADURA	ARRIBA COMPONENTES

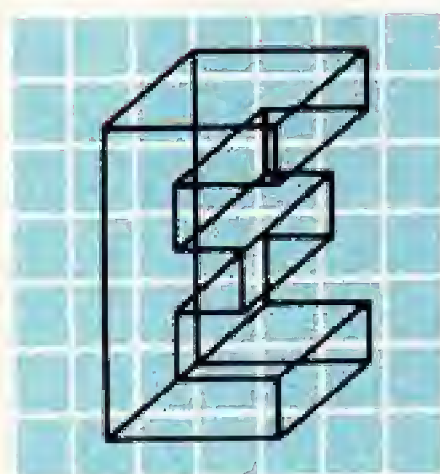
AMSTRAD CPC-64

	B	A
1	⊥	SONIDO
2	A14	A15
3	A12	A13
4	A10	A11
5	A8	A9
6	A6	A7
7	A4	A5
8	A2	A3
9	A0	A1
10	D6	D7
11	D4	D5
12	D2	D3
13	D0	D1
14	-MREQ	+5V
15	-RFSH	-M1
16	-RD	-IORQ
17	-HALT	-WR
18	-NMI	-INT
19	-BUSACK	-BUSRQ
20	-BUSRESET	READY
21	-ROMEN	RESET
22	-RAMRD	ROM DIS
23	CURSOR	RAM DIS
24	-EXP	LIGHT PEN
25	CLOCK	⊥
	ABAJO SOLDADURA	ARRIBA COMPONENTES

COMMODORE C-64

	B	A	
A	⊥	⊥	1
B	-ROM H	+5V	2
C	-RESET	+5V	3
D	-NMI	-IRQ	4
E	S02	R/-W	5
F	A15	DOT CLOCK	6
H	A14	I/001	7
J	A13	-GAME	8
K	A12	-EX ROM	9
L	A11	I/O 2	10
M	A10	-ROM L	11
N	A9	BA	12
P	A8	-DMA	13
R	A7	D7	14
S	A6	D6	15
T	A5	D5	16
U	A4	D4	17
V	A3	D3	18
W	A2	D2	19
X	A1	D1	20
Y	A0	D0	21
Z	⊥	⊥	22
	ABAJO SOLDADURA	ARRIBA COMPONENTES	

APRENDER CON EL ORDENADOR



L objeto de esta sección es el aprendizaje, de una manera amena, de una serie de conceptos que se estudian en la escuela. Para ello se utiliza la ayuda de los ordenadores, que son un vehículo de conocimiento absolutamente apasionante. La sección se subdivide a su vez en una parte dedicada a las aplicaciones de los ordenadores en las áreas de Naturaleza y Tecnología y en otra especializada en temas de Sociedad. En la primera parte se tratarán problemas matemáticos, químicos, físicos, biológicos, etc. En la segunda se abordarán temas lingüísticos, idiomáticos, geográficos. En ambas secciones existirán programas que sirvan para examinar sus conocimientos. Por último, existirá una parte dedicada a los más jóvenes de la casa, en la que se verán programas de aplicaciones sencillas.

NATURALEZA Y TECNOLOGIA

Problemas matemáticos

Cálculo de PI por el método de Montecarlo

El cálculo de PI no es algo convencional. Como sabemos, PI es un número irracional, lo cual significa que es imposible conocer con absoluta precisión su valor exacto, dado que consta de infinitas cifras decimales. Por ello se desarrollaron una serie de técnicas de aproximación para su cálculo, mediante las cuales podremos llegar a conocer con un cierto grado de exactitud su valor.

Una de las técnicas más exactas en su cálculo es el conocido como método de "Monte-Carlo", llamado así porque se basa en la generación aleatoria de puntos, simulando la actuación de una ruleta o el de unos dados.

El funcionamiento de este método es el siguiente:

En primer lugar, se toma un cuadrante de circunferencia cuyo radio sea la unidad. Por este motivo el área de dicho cuadrante será:

$$\text{Area} = \frac{\pi R^2}{4} = \frac{\pi \cdot 1^2}{4} = \frac{\pi}{4}$$

Podemos circunscribir el cuadrante de circunferencia en un cuadrado de lado 1. Su área valdrá uno. El programa se encargará de situar al azar puntos dentro de este cuadrado. Algunos de los puntos entrarán dentro del cuadrante de circunferencia, mientras que otros quedarán fuera.

La relación entre los puntos que estén dentro del cuadrante de la circunferencia y los que estén fuera será aproximadamente igual a la relación entre el área del cuadrante y la del cuadrado, sobre todo al aumentar el

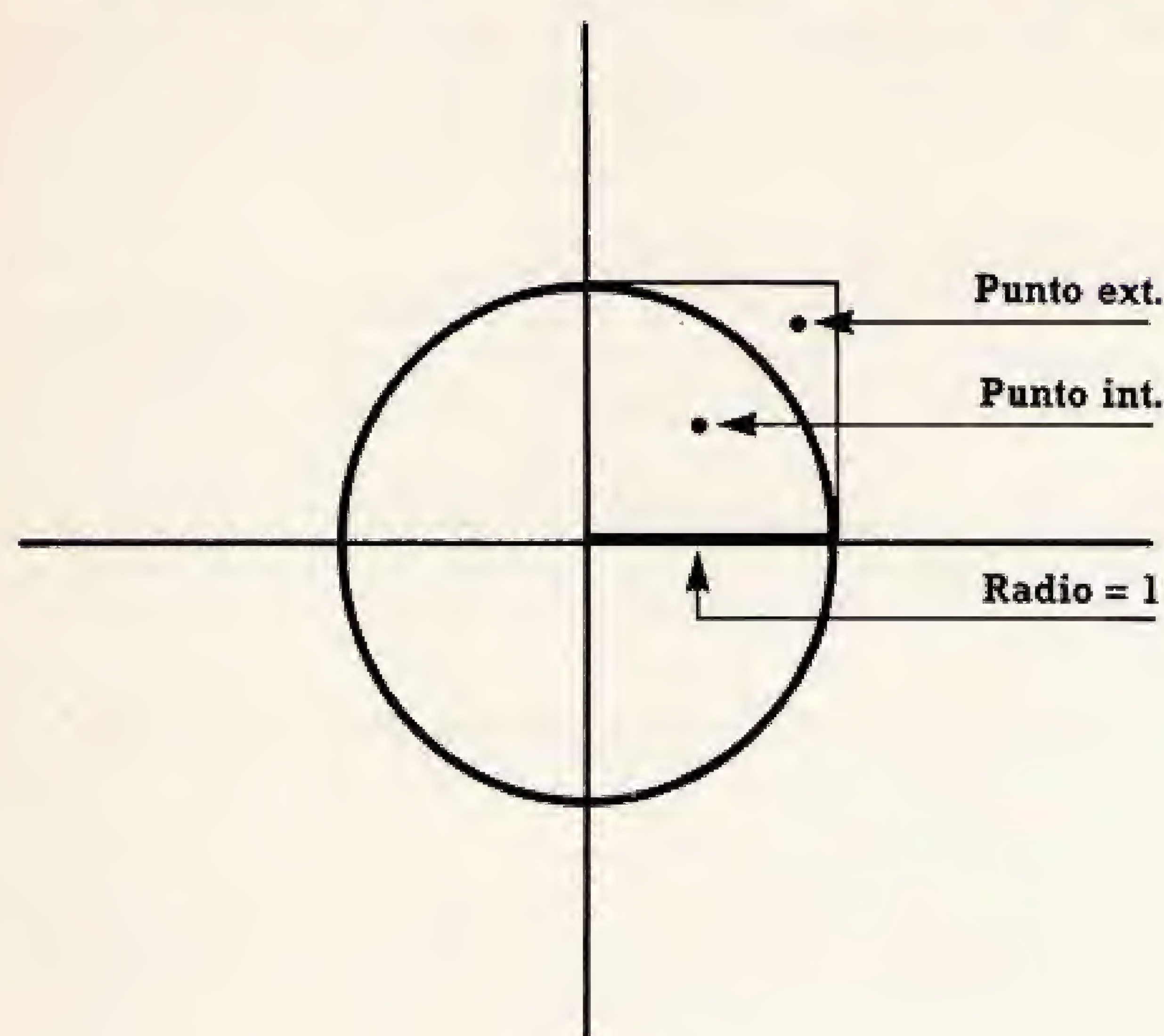


Fig. 1.—Puntos exteriores al cuadrante de la circunferencia.

número de puntos. Por tanto, podemos establecer la siguiente aproximación:

$$\frac{\frac{\pi}{4}}{1} = \frac{Ni}{Ne}$$

donde Ni = Número de puntos interiores.
Ne = Número de puntos exteriores.

Es decir:

$$\pi = 4 \cdot \frac{Ni}{Ne}$$

Cuanto mayor sea el número de puntos que situemos aleatoriamente, mayor será la exactitud del método.

```
10 REM ***********************************************
20 REM *                                             *
30 REM * CALCULO DE PI POR EL METODO DE MONTE-CARLO  *
40 REM *                                             *
50 REM ***********************************************
60 PI = 3.141592654#
70 REM
80 REM * DIBUJO DE LA PANTALLA *
90 REM
100 SCREEN 1,0
110 CLS
120 LOCATE 1,10
130 PRINT "* CALCULO DE PI *"
140 LINE (30,20)-(130,120),3,B
150 CIRCLE (30,120),100,3,0,PI/2,1
160 LOCATE 6,19
170 PRINT "Punto num. :"
180 LOCATE 10,19
190 PRINT "Puntos int.:";
200 LOCATE 14,19
210 PRINT "PI :"
220 REM
230 REM * INICIALIZACION DE VARIABLES *
240 REM
250 RANDOMIZE TIMER
260 LOCATE 20,1
270 INPUT "numero de puntos ->",N
280 DENTRO=0
290 REM
300 REM * COMIENZO DEL BUCLE PRINCIPAL *
310 REM
320 FOR I=1 TO N
330 X=RND:Y=RND       'CALCULO DE LAS COORDENADAS DEL PUNTO
340 DIS=X^2+Y^2
350 IF DIS<=1 THEN DENTRO=DENTRO+1:C=2 ELSE C=1
360 PSET((X*100+30),(120-Y*100)),C
370 LOCATE 6,31
380 PRINT I;
390 LOCATE 10,31
400 PRINT DENTRO
410 LOCATE 14,24
420 PRINT USING "#.######";DENTRO/I* 4
430 NEXT I
440 REM
450 REM * IMPRESION DEL VALOR CALCULADO FINAL *
460 REM
470 LOCATE 22,1
480 PRINT "El valor calculado de PI es :";DENTRO/N*4
```

Este programa es válido para IBM y compatibles.

SPECTRUM: **Todas las asignaciones han de ir precedidas del comando 'LET'.**
Quitar la línea 100
En donde aparezca 'LOCATE Y,X' cambiarlo por 'PRINT AT Y,X;'. X e Y pueden ser números o variables.
Línea 140 PLOT 30,20:DRAW 100,0:DRAW 0,100:DRAW -100,0:DRAW 0,-100
Línea 150 CIRCLE 30,20,100
Línea 250 RANDOMIZE
Línea 360 PLOT (X*100+30),(120-Y*100)
Línea 420 PRINT DENTRO/I*4
Línea 470 PRINT AT 21,1;

AMSTRAD: **Línea 100 MODE 1**
Línea 140 PSET (30,20):DRAWR (100,0):DRAWR (0,100):DRAWR (-100,0):DRAWR (0,-100)
Quitar la línea 150
Línea 250 RANDOMIZE
Línea 360 PLOT ((X*100+30),(120-Y*100))

La versión para MSX aparecerá en un próximo tomo.



Comentario del programa

El programa de cálculo de PI por el método de Montecarlo realiza la función anteriormente descrita. Genera una serie de números de coordenadas X,Y comprendidas entre 0 y 1 y comprueba si el punto está dentro del cuadrante de circunferencia. Esto lo hace calculando la distancia del punto generado al origen y si ésta es menor o igual a 1 resultará que el punto será interior.

Se ha utilizado un BASIC estándar en el programa, salvo, lógicamente, en el caso de los comandos gráficos. Si quisiéramos aumentar la velocidad del programa, podríamos omitirlos. De esta forma la exactitud en el cálculo de PI podría ser mayor, dado que tendríamos la capacidad de aumentar el número de puntos generados al azar sin que ello supusiese un tiempo de ejecución del programa demasiado largo.

Descomposición en factores primos

Al efectuar la división de un número a cualquiera por otro número **b** cualquiera, puede suceder lo siguiente:

— El resto de la división es igual a cero. En ese caso la división es exacta y b es **divisor** de a.

— El resto de la división es distinto de cero. La división es inexacta.

```
 8  |2
(0)  4      DIVISION EXACTA

 9  |2
(1)  4      DIVISION INEXACTA
```

Fig. 2.—Tipos de divisiones de un número natural.

En general puede decirse que un número divide a otro cuando el resto de la división es igual a cero. El resto de una división puede considerarse como el resultado de restar del dividendo el producto del divisor por el cociente. Por otra parte, se denomina división entera al resultado de obtener el cociente de un número sin decimales. Por tanto, puede decirse que el cociente de una división equivale al resultado de una división entera.

```
96  |8
 4  (9)     = INT (96/8)
```

Fig. 3.—El cociente de una división equivale a una división entera.

Cuando en lenguaje BASIC dividimos un número por otro no obtenemos un resultado entero. Si quisiéramos obtener el cociente tendríamos que efectuar la división entera. Ello se expresaría del siguiente modo:

COCIENTE = INT (DIVIDENDO/DIVISOR)

donde INT es una función que calcula la parte entera de un número, es decir, el número sin decimales, por tanto, el cociente.

Por otra parte, se denomina número **primo** a aquél que sólo es divisible por sí mismo y por la unidad. Ejemplos de números primos son, por ejemplo, el 2, el 13, el 29, etc.

En este caso vamos a factorizar un número. Es decir, vamos a expresar un número cualquiera como producto de factores primos. Para ello se toma un número y se empieza a dividir por el primer factor primo diferente de 1, esto es, el 2. Si el número es divisible por 2, éste es un factor primo del número. A continuación se toma el cociente de la división anterior y se divide de nuevo por 2. Se sigue este mismo proceso hasta que el número no sea divisible por 2, en cuyo caso se repiten todos los cálculos para el siguiente factor, que es el 3.

Este es el procedimiento que sigue el programa. El organigrama correspondiente sería el siguiente:

```
DESCOMPOSICION DEL NUMERO  5664

        5664      ! 2
        2832      ! 2
        1416      ! 2
        708       ! 2
        354       ! 2
        177       ! 3
        59        ! 59
        1


   5664 = 2 ^ 5 * 3 ^ 1 * 59 ^ 1 * 1
```

Fig. 4.—Descomposición en factores primos de un número cualquiera.

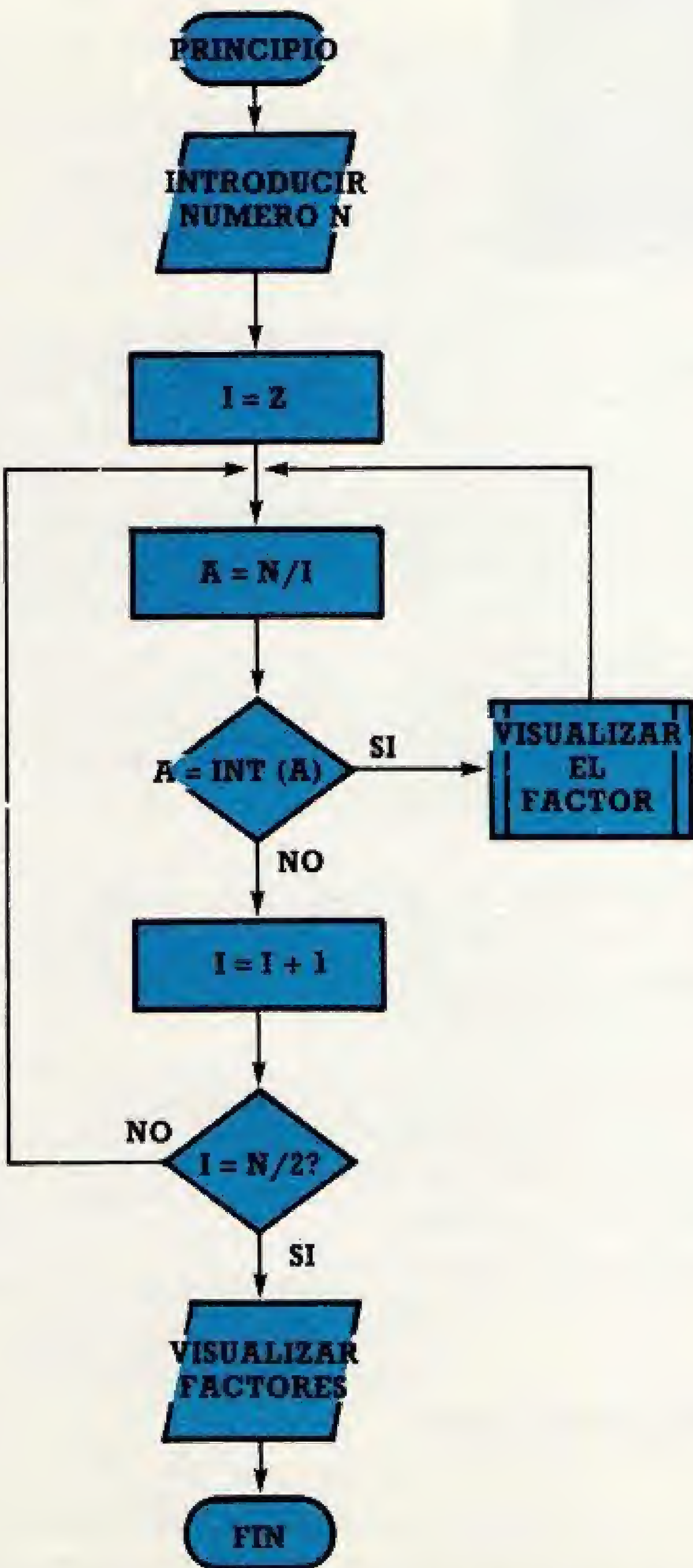


Fig. 5.—Organigrama del programa de descomposición en factores primos de un número.

Ello se traduciría en el siguiente programa:

```
10 REM *******************************
20 REM * DESCOMPOSICION DE UN NUMERO *
30 REM *     EN FACTORES PRIMOS      *
40 REM *******************************
50 CLS
60 REM * ENTRADA DE DATOS *
70 INPUT "QUE NUMERO QUIERES DESCOMPONER";N
80 CLS
90 PRINT "DESCOMPOSICION DEL NUMERO ";N
100 PRINT:PRINT
110 DIM S(N/2)
120 LET M=N
130 REM * PROCESO DE FACTORIZACION *
140 FOR I=2 TO N/2
150 LET A=N/I
160 IF A=INT(A) THEN GOSUB 500:GOTO 150
170 IF A<=1 THEN GOTO 190
180 NEXT I
190 PRINT TAB(10);1
200 PRINT:PRINT:PRINT:PRINT
210 PRINT M;"=";
220 REM * AGRUPACION DE FACTORES COMUNES *
230 FOR I=2 TO M/2
240 IF S(I)<>0 THEN GOSUB 1000
250 NEXT I
260 PRINT 1
270 END
500 REM *************************
510 REM * VISUALIZACION PARCIAL *
520 REM *************************
530 PRINT TAB(10);N;TAB(20);"!";I
540 LET S(I)=S(I)+1
550 LET N=A
560 RETURN
1000 REM ***********************
1010 REM * VISUALIZACION FINAL *
1020 REM ***********************
1030 PRINT I;"^";S(I);"*";
1040 RETURN
```

Este programa es válido para IBM, MSX, AMSTRAD Y SPECTRUM.

MSX:	**Cambiar las sentencias 'LOCATE Y,X' por 'LOCATE X,Y'.**
COMMODORE:	**Para simular la sentencia 'LOCATE' utiliza el programa que se da en la sección de TRUCOS DE PROGRAMACION.** **Línea 80 PRINT CHR$(147)**

El programa consta de dos partes:

— En una primera fase se realiza el proceso de factorización visualizando los resultados parciales.

— En la segunda se efectúa la visualización del número original descompuesto en factores primos.

Para agrupar todos los factores comunes del número se ha utilizado una variable con subíndice denominada genéricamente S (N/2), siendo N el número que se desea descomponer.

PASATIEMPOS MATEMATICOS

Binomio de Newton y triángulo de Tartaglia

El desarrollo de la potencia natural de un binomio sigue la fórmula de Newton:

$$(a + b)\hat{}N = \binom{N}{0} a^{n} \cdot b^{a} + \binom{N}{1} a^{n-1} \cdot b^{1} + \binom{N}{2} + \ldots$$

$$\ldots + \binom{N}{N-2} a^{2} \cdot b^{n-2} + \binom{N}{N-1} a^{1} \cdot b^{n-1} + \binom{N}{N} a^{0} \cdot b^{n}$$

Fig. 6.

donde

$$\binom{N}{0}, \binom{N}{1} \quad \binom{N}{N-2}, \binom{N}{N-1}, \binom{N}{N}$$

Fig. 7.

son números combinatorios cuya fórmula es la siguiente:

$$\binom{N}{m} = \frac{N!}{(N-m)! \cdot m!}$$

Fig. 8.

Si el número de sumandos es relativamente grande, el cálculo de todos y cada uno de los elementos puede ser una labor tediosa. Existe un método alternativo que permite obtener los coeficientes del binomio de Newton de una manera sencilla. Para ello vamos a construir un triángulo de valores numéricos que se denomina triángulo de Tartaglia. Su elaboración se basa en la siguiente propiedad de los números combinatorios:

$$\binom{N}{m} = \binom{N-1}{m} + \binom{N-1}{m-1}$$

Fig. 9.

De esta forma se puede expresar un número combinatorio en función de dos que le precedan. El triángulo de Tartaglia sería:

```
                    1
                   1 1
                  1 2 1
                 1 3 3 1
                1 4 6 4 1
              1 5 10 10 5 1
             1 6 15 20 15 6 1
           1 7 21 35 35 21 7 1
          1 8 28 56 70 56 28 8 1
        1 9 36 84 126 126 84 36 9 1
     1 10 45 120 210 252 210 120 45 10 1

EL DESARROLLO DEL BINOMIO ES:

(a+b)^ 10 = 1 *(a^ 10 *b^ 0 )+ 10 *(a^ 9 *b^ 1 )+ 45 *(a^ 8 *b^ 2 )+ 120 *(a^ 7
*b^ 3 )+ 210 *(a^ 6 *b^ 4 )+ 252 *(a^ 5 *b^ 5 )+ 210 *(a^ 4 *b^ 6 )+ 120 *(a^ 3
*b^ 7 )+ 45 *(a^ 2 *b^ 8 )+ 10 *(a^ 1 *b^ 9 )+ 1 *(a^ 0 *b^ 10 )
Ok
```

Fig. 10.—Triángulo de Tartaglia y binomio de Newton de la décima potencia.

Cada elemento del Triángulo está definido por un número de fila y uno de columna. Un elemento será igual al elemento de la fila anterior, con igual número de columna, más el elemento de la fila anterior con el número de columna anterior. Por ejemplo, el 10, que está en la sexta fila y en la tercera columna, es el resultado de sumar el 6 que está en la quinta fila y la tercera columna y el 4 que está en la quinta fila y en la segunda columna.

Cada una de las filas del triángulo se corresponde con los coeficientes de uno de los desarrollos de Newton. Los coeficientes de un binomio elevado a una potencia "n" serán los números de la fila "n" del triángulo de Tartaglia.

El método de resolución del binomio de Newton mediante el Triángulo de Tartaglia es muy sencillo de convertir en programa en Basic. Se trata de hacer sumas sucesivas para calcular cada fila en función de la anterior. El algoritmo es muy sencillo, ya que refleja de modo natural el método de generación del triángulo.

El programa se ha realizado en un BASIC lo más estándar posible. Las sentencias de tipo gráfico son propias del ordenador Spectrum, por lo cual habría que adaptarlas a cada ordenador.

```
10 REM ***************************
20 REM *        RESOLUCION       *
30 REM *            DEL          *
40 REM *    BINOMIO DE NEWTON    *
50 REM *          MEDIANTE       *
60 REM *             EL          *
70 REM * TRIANGULO DE TARTAGLIA *
80 REM ***************************
90 DIM TR(20)
100 TR(1)=1:LP=80
110 CLS
120 REM *******************
130 REM * MENSAJE INICIAL *
140 REM *******************
150 LOCATE 10,10:PRINT"ESTE PROGRAMA CALCULA LOS COEFICIENTES DEL BINOMIO"
160 LOCATE 11,10:PRINT"DE NEWTON BASANDOSE EN EL TRIANGULO DE TARTAGLIA."
170 LOCATE 13,10:PRINT"LOS COEFICIENTES DEL BINOMIO SON LOS NUMEROS QUE"
180 LOCATE 14,10:PRINT"APARECEN EN LA BASE DEL TRINAGULO GENERADO."
190 LOCATE 20,10: INPUT "INTRODUCE EL EXPONENTE (a+b)^",N
200 IF N>16 THEN LOCATE 18,10:PRINT "DEMASIADO GRANDE":GOTO 190
210 REM **************************
220 REM * TRIANGULO DE TARTAGLIA *
230 REM **************************
240 CLS
250 PRINT:PRINT TAB(38);1
260 FOR I=1 TO N
270     TEMP=1
280     FOR J=2 TO I
290         VALOR=TR(J)+TEMP
300        TEMP=TR(J)
310        TR(J)=VALOR
320    NEXT J
330    TR(I+1)=1
340 REM ****************************
350 REM * SITUACION EN LA PANTALLA *
360 REM ****************************
370    FI$=""
380    FOR J=1 TO I+1
390        FI$=FI$+STR$(TR(J))
400    NEXT J
410    PRINT TAB((LP-LEN(FI$))/2);FI$
420 NEXT I
430 REM *********************
440 REM *   VISUALIZACION   *
450 REM *        DEL        *
460 REM * BINOMIO DE NEWTON *
470 REM *********************
```

```
480 PRINT
490 PRINT"EL DESARROLLO DEL BINOMIO ES:"
500 PRINT
510 PRINT"(a+b)^";N;"=";
520 FOR I=0 TO N
530   PRINT TR(I+1);"*(a^";N-I;"*b^";I;")";
540   IF I<>N THEN PRINT"+";
550 NEXT I
```

Este programa es válido para IBM, MSX y AMSTRAD.

COMMODORE: **Las variables no pueden tener un nombre de más de dos caracteres. Por ejemplo, si una variable se llama VALOR, la cambiaríamos por VA. Sustituir los comandos CLS por PRINT CHR$(147) Para simular la sentencia locate utiliza la rutina que se da en la sección de TRUCOS DE PROGRAMACION.**

SPECTRUM: **Las variables dimensionadas no pueden tener más de un caracter como nombre. As:i, la matriz TR() habrá que denominarla T(). Sustituir todos los LOCATE por PRINT AT. Las variables alfanuméricas no pueden tener como nombre más de un caracter. Así, la variable alfanumérica FI$ la cambiaremos por F$.**

SIMULACION DE FENOMENOS

Simulación de la Ley de Gay Lussac

Es de todos conocido que un recipiente, como puede ser un émbolo cuyas paredes no sean rígidas, experimenta un aumento de volumen al aumentar la temperatura. Esta sería una forma de enunciar la ley de Gay Lussac. En este fenómeno intervienen dos variables, que son la temperatura y el volumen. La presión se mantiene constante. La expresión matemática sería la siguiente:

$$V / T = V' / T'$$

Esta ley sirvió como base a la elaboración de la ley de los gases perfectos, cuya fórmula es la siguiente:

$$pV = nRT$$

donde

p es la presión
V el volumen
n el número de moles
R la constante de Avogadro
T la temperatura

El programa es una simulación sencilla de la ley de Gay Lussac. En él se dibuja un termómetro, un émbolo y una gráfica. Si se pulsa una de las teclas de movimiento del cursor arriba o abajo se observará que al pulsar la flecha hacia arriba el émbolo experimenta una expansión, es decir, un aumento de volumen. Al mismo tiempo el mercurio del termómetro también se expande, lo cual se traduce en un aumento de temperatura. Estos cambios se producen en sentido contrario al pulsar la flecha hacia abajo.

Todo ello se refleja en una representación gráfica. Se obtiene una recta, lo cual era previsible, dado que la relación entre el aumento de volumen y el de la temperatura a presión constante y sin que se experimente una variación en el número de moles, es un valor constante.



Comentario del programa

```
10 REM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
20 REM %                               %
30 REM % SIMULACION DEL COMPORTAMIENTO %
40 REM %                               %
50 REM % DE UN GAS A PRESION CONSTANTE %
60 REM %                               %
70 REM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80 MODE 1:CLS:CLG
90 PEN 1
100 LOCATE 15,5
110 FOR i=1 TO 20:PRINT CHR$(143);:NEXT i
120 LOCATE 15,11
130 FOR i=1 TO 20:PRINT CHR$(143);:NEXT i
140 FOR i=1 TO 7:LOCATE 35,i+4:PRINT CHR$(143);:NEXT i
150 MOVE 208,64:DRAW 558,64,1:MOVE 224,48:DRAW 224,206
160 MOVE 28,48:DRAW 28,332:MOVE 50,48:DRAW 50,332
170 DRAWR -10,10:DRAWR -2,0:DRAWR -10,-10
180 PEN 3
190 LOCATE 2,23:PRINT CHR$(214)+CHR$(143)+CHR$(215)
200 LOCATE 2,24:PRINT CHR$(213)+CHR$(143)+CHR$(212)
210 PEN 1
220 FOR I = 1 TO 18:LOCATE 5,i+4:PRINT"-":NEXT i
230 j=0
240 FOR i=85 TO 0 STEP -5
250 LOCATE 6,5+j:PRINT i;
260 j=j+1:NEXT i
270 LOCATE 35,22:PRINT"t";
280 LOCATE 14,13:PRINT"v";
290 LOCATE 1,1:PRINT"USA LAS FLECHAS PARA AUMENTAR O REDUCIR"
300 LOCATE 3,2:PRINT"LA TEMPERATURA Y OBSERVA EL VOLUMEN"
310 MOVE 128,368:DRAWR 0,-16:DRAWR -64,0:DRAWR 0,-16
320 MOVE 560,368:DRAWR 0,-16:DRAWR -64,0:DRAWR 0,-16
330 LOCATE 15,24:PRINT"PULSA 'S' PARA ACABAR";
340 REM
350 REM %% Inicializacion de variables %%
360 REM
370 PE=34:PEN 2:E$=CHR$(143):GOSUB 1170:PEN 1:E$=CHR$(207):PE=33:GOSUB 1170
380 T=0:PT=22:PEN 3:LOCATE 3,PT:PRINT CHR$(143);:PEN 1
390 MOVE 224,72:PLOTR 0,0,3
400 REM %% Bucle principal %%%
410 A$=INKEY$
420 IF A$=CHR$(240) AND T<85 THEN T=T+5: PT=PT-1:GOSUB 1000
430 IF A$=CHR$(241) AND T>0 THEN GOSUB 1000:T=T-5: PT=PT+1
440 IF UPPER$(a$)="S" THEN GOTO 490
450 GOTO 410
460 REM
470 REM %% Fin de bucle principal %%%%%%%%%%%%%%%
480 REM
490 CLS:CLG
500 LOCATE 5,2:PRINT"Como habras podido observar al"
510 LOCATE 5,3:PRINT"aumentar la temperatura del gas"
520 LOCATE 5,4:PRINT"el  volumen de este aumenta, y"
530 LOCATE 5,5:PRINT "al disminuir la temperatura el "
540 LOCATE 5,6:PRINT "volumen disminuye."
550 LOCATE 5,7:PRINT "      Por lo tanto la variacion "
560 LOCATE 5,8:PRINT "del volumen es directamente pro-"
570 LOCATE 5,9:PRINT"porcional a la variacion de tem-"
580 LOCATE 5,10:PRINT"peratura, siempre que esta ocurra"
590 LOCATE 5,11:PRINT"a presion constante."
600 LOCATE 5,13:PRINT"    La formula que establece la "
610 LOCATE 5,14:PRINT"relacion entre estas dos magnitu-"
620 LOCATE 5,15:PRINT"des se denomina LEY DE GAY-LUSSAC,"
630 LOCATE 5,16:PRINT"y su expresion matematica es:"
640 LOCATE 10,18:PRINT" V      V' "
650 LOCATE 10,19:PRINT"--- = ---"
660 LOCATE 10,20:PRINT" T      T'"
670 END
900 REM
950 REM %% Subrutina de actualizacion %%%%%%%%%%%%
970 REM
1000 IF A$=CHR$(240) THEN D$=CHR$(143) ELSE D$=" "
1010 PEN 3:LOCATE 3,PT:PRINT D$;:PEN 1
1020 IF A$=CHR$(240) THEN GOSUB 1090 ELSE GOSUB 1130
1030 IF A$=CHR$(240) THEN INX=16:INY=8 ELSE INX=-16:INY=-8
1040 DRAWR INX,INY,1:PLOTR 0,0,3:PEN 1
1050 RETURN
1060 REM
1070 REM %% Subrutina de actualizacion del volumen  %%%%%%%%
```

```
1080 REM
1090 E$=CHR$(143):PEN 2:GOSUB 1170
1100 PE=PE-1
1110 E$=CHR$(207):PEN 1:GOSUB 1170
1120 RETURN
1130 E$=" ":GOSUB 1170
1140 PE=PE+1
1150 E$=CHR$(207):PEN 1:GOSUB 1170
1160 RETURN
1170 FOR I=1 TO 5:LOCATE PE,I+5
1180 PRINT E$: NEXT I
1190 RETURN
```

Válido para AMSTRAD. El resto de las versiones irán apareciendo en futuros tomos.

PRUEBA TUS CONOCIMIENTOS

Test de geografía

Aprender geografía supone el conocer los lugares en los que nos encontramos. Aunque parezca mentira, no todo el mundo conoce la localización exacta de las ciudades de España. Este conocimiento es importante, por ejemplo, para realizar viajes por la Península.

El programa pretende realizar un pequeño test de las ciudades de España. Para ello se dibuja un mapa de la Península en el cual aparecerá una de las ciudades iluminadas, de la cual es preciso determinar su nombre. Aparecen cinco posibilidades de las cuales sólo una es correcta.

```
10 REM ***************************
20 REM **  TEST DE GEOGRAFIA    **
30 REM **                       **
40 REM ** MAPA DE LA PENINSULA **
50 REM **                       **
60 REM ***************************
70 MODE 1:CLG
80  LOCATE 1,9: PRINT "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
90 LOCATE 1,10: PRINT "XXXX                                XXXX"
100 LOCATE 1,11: PRINT "XXXX           GEOGRAFIA           XXXX"
110 LOCATE 1,12: PRINT "XXXX                                XXXX"
120 LOCATE 1,13: PRINT "XXXX       CIUDADES DE ESPANA      XXXX"
130 LOCATE 1,14: PRINT "XXXX                                XXXX"
140 LOCATE 1,15: PRINT "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
150 FOR X=1 TO 1000: NEXT X
160 CLS
170 REM ********************
180 REM **  DIBUJAR MAPA  **
190 REM ********************
200 LOCATE 10,2: PRINT "CIUDADES DE ESPANA"
210 DIM a(47): DIM b(47): DIM a$(47): DIM b$(4): DIM c$(47)
220 INK 3,20:INK 2,6,20
230 ORIGIN 8,56
240 PLOT 99,8,1
250 FOR x=1 TO 101
260 READ c,d
270 DRAW c,d,1
280 NEXT x
290 REM ******************************
300 REM ** PONER PUNTOS DE CIUDADES **
310 REM ******************************
320 FOR x=1 TO 47
330 READ c,d:LET a(x)= INT (c*1.6): LET b(x)= INT (d*1.6)
340 PLOT a(x),b(x),3:PLOTR 1,0:PLOTR 0,1:PLOTR -1,0
350 NEXT x
360 REM ***************************
370 REM **  LEER NOMBRES CIUDADES **
380 REM ***************************
390 FOR x=1 TO 47
400 READ a$(x):LET c$(x)=a$(x)
410 NEXT x
420 REM *********************
430 REM **  ESCRIBIR TEXT  **
440 REM *********************
450 LET aa=47
460 LET au=0:LET av=0
```

```
470 FOR x=1 TO 4: LET b$(x)="":NEXT x
480 LET ab=1+INT(RND*aa): LET ac=1+INT(RND*4)
490 LET al=1
500 IF a$(ab)=c$(al) THEN GOTO 520
510 LET al=al+1: GOTO 500
520 LET b$(ac)=a$(ab)
530 LET x=ab:PLOT a(X)+1,b(x),2:GOSUB 870:PEN 1
540 FOR x=1 TO 4
550 IF x=ac THEN GOTO 630
560 IF al=47 THEN GOTO 600
570 IF au=0 THEN LET b$(x)=c$(al+1):LET au=1:GOTO 630
580 IF al=1 THEN GOTO 600
590 IF av=0 THEN LET b$(x)=c$(al-1):LET av=1:GOTO 630
600 LET at=1+INT (RND*47):LET b$(x)=c$(at)
610 FOR w=1 TO x-1: IF b$(w)=b$(x) THEN GOTO 600
620 NEXT w
630 LOCATE 24,13+x: PRINT x;". ";b$(x)
640 NEXT x
650 REM ******************************
660 REM **  EXAMINAR CONTESTACION   **
670 REM ******************************
680 LOCATE 25,9: PRINT "CIUDAD?"
690 LET d$=INKEY$: IF d$="" THEN GOTO 690
700 LET aj=VAL (d$):IF aj<1 OR aj>4 THEN GOTO 690
710 IF aj=ac THEN LOCATE 24,20: PRINT "CORRECTO": GOTO 730
720 LOCATE 24,20: PRINT "INCORRECTO"
730 PEN 2: LOCATE 24,13+ac:PRINT ac;". ";b$(ac):PEN 1
740 LOCATE 5,24:PRINT "PULSE UNA TECLA PARA CONTINUAR"
750 IF INKEY$="" THEN GOTO 750
760 LOCATE 5,24:PRINT "
770 FOR x=5 TO 20
780 LOCATE 24,x: PRINT "
790 NEXT x
800  LET x=ab:PLOT a(x)+1,b(x),0: GOSUB 870:PLOT A(X),B(X),3: PEN 1
810 IF aa=1 THEN RESTORE 1100:GOTO 320
820 FOR x=ab TO aa-1
830 LET a$(x)=a$(x+1):LET a(x)=a(x+1):LET b(x)=b(x+1)
840 NEXT x
850 LET aa=aa-1
860 GOTO 460
870 PLOTR 0,3:PLOTR 3,0:PLOTR 0,-3:PLOTR -3,0
880 RETURN
890 REM ********************
900 REM ****    DATOS    ****
910 REM ********************
920 DATA 101,23,109,26,117,25,128,35,141,37,149,37
930 DATA 163,35,181,36,186,38,190,35,199,51,204,59
940 DATA 210,61,208,63,210,64,229,68,230,76,232,80
950 DATA 232,84,234,87,234,92,237,93,248,102,256,105
960 DATA 256,109,247,111,238,128,236,144,274,179
970 DATA 268,181,285,195,304,203,307,206,330,221
980 DATA 328,234,330,233,336,240,332,245,323,246
990 DATA 316,244,315,241,304,243,298,243,291,245
1000 DATA 290,249,283,250,281,250,261,254,262,250
1010 DATA 243,250,234,252,235,249,227,255,202,268
1020 DATA 198,268,196,272,182,268,167,272,163,269
1030 DATA 152,275,129,272,92,280,62,279,46,285
1040 DATA 45,288,29,279,30,274,27,275,13,274,0,262
1050 DATA 6,254,15,250,10,223,28,230,32,221
1060 DATA 62,224,71,224,71,212,77,212,77,208,60,189
1070 DATA 62,180,60,162,57,159,59,150,52,141,40,140
1080 DATA 48,126,56,116,46,108,45,99,53,89,52,84,37
1090 DATA 67,39,54,53,53,68,40,66,36,72,28,90,7,99,8
1100 DATA 24,148,12,152,16,171,30,164,58,170,61,153
1110 DATA 91,171,79,141,92,148,105,165,120,167,110,158
1120 DATA 114,148,125,157,139,132,147,143,164,130,201,143
1130 DATA 190,128,176,123,153,95,126,103,102,110,113,135
1140 DATA 85,115,75,132,56,129,58,118,45,84,75,129,36,72,92,105
1150 DATA 118,96,147,84,147,59,135,50,123,72,88,72
1160 DATA 36,70,33,37,46,18,50,38,73,48,78,23,93,32,90,46,113,25
1170 DATA ORENSE,PONTEVEDRA,LA CORUNA,LUGO,OVIEDO,LEON
1180 DATA SANTANDER,PALENCIA,BURGOS,BILBAO,S. SEBASTIAN,VITORIA
1190 DATA LOGRONO,PAMPLONA,ZARAGOZA,HUESCA,LERIDA,GERONA
1200 DATA BARCELONA,TARRAGONA,CASTELLON,TERUEL,GUADALAJARA,SORIA
1210 DATA SEGOVIA,VALLADOLID,ZAMORA,SALAMANCA,CACERES,AVILA,TOLEDO,MADRID
1220 DATA CUENCA,VALENCIA,ALICANTE,MURCIA,ALBACETE,CIUDAD REAL,BADAJOZ
1230 DATA HUELVA,CADIZ,SEVILLA,CORDOBA,MALAGA,GRANADA,JAEN,ALMERIA
```

Programa válido para AMSTRAD. El resto de las versiones irán apareciendo en futuros tomos.



SOCIEDAD

Programa de inglés

El inglés es un idioma que en nuestros días es imprescindible. No es un lenguaje complicado, pero, lógicamente, es necesario aprenderlo. Como todos los idiomas, posee una gramática propia. En cualquier caso, no es de las más complicadas. Una de las partes más sencillas y que lo diferencia de los demás idiomas es la sencillez de sus verbos. La formación de un tiempo de un verbo a partir del infinitivo es tan sencilla como añadir las letras "ed" al infinitivo del verbo en cuestión, en el caso de verbos regulares. En el caso del programa que nos ocupa tratamos con los verbos irregulares. Para manejarse mínimamente con el idioma, no queda más remedio que aprender las formas irregulares de los verbos. Ese es el objetivo del programa. Se han seleccionado una serie de verbos irregulares que se utilizan con frecuencia. El programa presenta en primer lugar, la lista completa de los verbos y luego pide que se introduzca la forma completa de uno de los verbos en cuestión. Lo único que debe hacerse es contestar correctamente.

```
10 REM ***********************
20 REM * PROGRAMA DE INGLES *
30 REM * VERBOS IRREGULARES *
40 REM ***********************
50 CLS
60 DIM V$(20):DIM P$(20):DIM S$(20)
70 DIM C(20)
80  REM *********************
90  REM * LECTURA DE DATOS *
100 REM *********************
110 FOR I=1 TO 20
120 READ V$(I),P$(I),S$(I)
130 NEXT I
140 REM *************************
150 REM * PRIMERA VISUALIZACION *
160 REM *************************
170 PRINT TAB(10);" LEE ATENTAMENTE "
180 PRINT TAB(10);" LA SIGUIENTE TABLA"
190 PRINT TAB(10);" LUEGO TENDRAS QUE RESPONDER"
200 PRINT :PRINT "ASEGURATE QUE TU TECLADO ESTA EN MAYUSCULAS"
210 PRINT:PRINT "PULSA CUALQUIER TECLA PARA CONTINUAR"
220 LET A$=INKEY$:IF A$="" THEN GOTO 220
230 CLS
240 FOR I=1 TO 20
250 PRINT V$(I);TAB(10);P$(I);TAB(20);S$(I)
260 NEXT I
270  FOR I=1 TO 5000:NEXT I
280 REM ************
290 REM * PREGUNTA *
300 REM ************
310 CLS
```

```
320 RANDOMIZE TIMER
330 FOR J=1 TO 20
340 CLS
350 LET I=INT(RND*20)+1
360 IF C(I)>0 THEN GOTO 350
370 LET C(I)=C(I)+1
380 PRINT "EL INFINITIVO DEL VERBO ES:";V$(I)
390  PRINT "INTRODUCE LOS TRES TIEMPOS"
400 INPUT V$,P$,S$
410 IF V$=V$(I) AND P$=P$(I) AND S$=S$(I) THEN GOSUB 460:GOTO 430
420 GOSUB 540
430 PRINT "PULSA UNA TECLA PARA CONTINUAR"
440 LET A$=INKEY$:IF A$="" THEN 440
450 NEXT J
460 REM ************
470 REM * CORRECTO *
480 REM ************
490 PRINT " C O R R E C T O "
500 LET A=A+1
510 PRINT "ACIERTOS:";A
520 PRINT "FALLOS:";F
530 RETURN
540 REM **************
550 REM * INCORRECTO *
560 REM **************
570 PRINT " I N C O R R E C T O "
580 PRINT:PRINT " LA SOLUCION ERA"
590 PRINT V$(I);",";P$(I);",";S$(I)
595 LET F=F+1
600 PRINT "ACIERTOS:";A
610 PRINT "FALLOS:";F
620 RETURN
630 REM *********
640 REM * DATOS *
650 REM *********
660 DATA CUT,CUT,CUT
670 DATA DRINK,DRANK,DRUNK
680 DATA FALL,FELL,FALLEN
690 DATA FIND,FOUND,FOUND
700 DATA GO,WENT,GONE
710 DATA KEEP,KEPT,KEPT
720 DATA KNOW,KNEW,KNOWN
730 DATA LEAVE,LEFT,LEFT
740 DATA LET,LET,LET
750 DATA MAKE,MADE,MADE
760 DATA PAY,PAID,PAID
770 DATA PUT,PUT,PUT
780 DATA READ,READ,READ
790 DATA RUN,RAN,RUN
800 DATA SEE,SAW,SEEN
810 DATA SET,SET,SET
820 DATA SLEEP,SLEPT,SLEPT
830 DATA TAKE,TOOK,TAKEN
840 DATA TELL,TOLD,TOLD
850 DATA WRITE,WROTE,WRITTEN
```

Válido para MSX, IBM, AMSTRAD y COMMODORE.

COMMODORE: Línea 220 GET A$:IF AS="" THEN GOTO 220
Línea 440 GET A$:IF A$="" THEN GOTO 440

SPECTRUM: Línea 320 RANDOMIZE
Línea 60 DIM V$(20,10):DIM P$(20,10):DIM S$(20,10)

```
CUT      CUT      CUT
DRINK    DRANK    DRUNK
FALL     FELL     FALLEN
FIND     FOUND    FOUND
GO       WENT     GONE
KEEP     KEPT     KEPT
KNOW     KNEW     KNOWN
LEAVE    LEFT     LEFT
LET      LET      LET
MAKE     MADE     MADE
PAY      PAID     PAID
PUT      PUT      PUT
READ     READ     READ
RUN      RAN      RUN
SEE      SAW      SEEN
SET      SET      SET
SLEEP    SLEPT    SLEPT
TAKE     TOOK     TAKEN
TELL     TOLD     TOLD
WRITE    WROTE    WRITTEN
```

Fig. 11.—Tabla de verbos irregulares ingleses.

```
EL INFINITIVO DEL VERBO ES:MAKE
INTRODUCE LOS TRES TIEMPOS
? MAKE,MADE,MADE
 C O R R E C T O
ACIERTOS: 1
FALLOS: 0
PULSA UNA TECLA PARA CONTINUAR
```

Fig. 12.—Test de inglés.

PARA LOS MAS JOVENES

El programa siguiente trata de que los más pequeños tengan la oportunidad de probar sus conocimientos matemáticos frente al ordenador. Consiste en un programa que genera aleatoriamente dos sumandos. Lo único que hay que hacer es determinar la suma.

```
7151    7151    7151    7151    7151    7151
6831    6831    6831    6831    6831    6831
----    ----    ----    ----    ----    ----
   ?     ?2     ?82    ?982   ?3982   13982

ACERTASTE
 DESEAS CONTINUAR
```

Fig. 13.—Ejecución del programa de sumas.

```
10 REM *********************
20 REM * PROGRAMA DE SUMAS *
30 REM *********************
40 CLS
50 REM * GENERACION DE NUMEROS *
60 LET A=INT(RND*9999)+1
70 LET B=INT(RND*9999)+1
80 LET S=A+B
90 LET S$=STR$(S)
100 LET L=LEN(S$)
110 REM * PANTALLA DE SUMAS *
120 LOCATE 10,10:PRINT USING "####";A
130 LOCATE 11,10:PRINT USING "####";B
140 LOCATE 12,10:PRINT "----"
150 FOR I=1 TO L-1
160 LOCATE 13,14-I
170 PRINT "?"
180 A$(I)=INKEY$:IF A$(I)="" THEN GOTO 180
190 A(I)=VAL(A$(I))
200 LOCATE 13,14-I:PRINT A$(I)
210 NEXT I
220 REM * CALCULO DE RESULTADO *
230 LET SUMA=0
240 FOR I=1 TO L-1
250 LET C=A(I)*10^(I-1)
260 LET SUMA=SUMA+C
270 NEXT I
280 REM * RESULTADO FINAL *
290 IF SUMA=S THEN LOCATE 20,10:PRINT "ACERTASTE"
300 IF SUMA<>S THEN LOCATE 20,10:PRINT "LASTIMA"
310 REM * MAS SUMAS *
320 PRINT "DESEAS CONTINUAR"
330 LET A$=INKEY$:IF A$="" THEN GOTO 330
340 IF A$="S" OR A$="s" THEN GOTO 40
350 END
```

Válido para MSX, IBM, AMSTRAD, COMMODORE y SPECTRUM.

Nota:
Ver comentarios generales de los demás programas y aplicarlos en éste.

Tabla de conversión para diferentes ordenadores

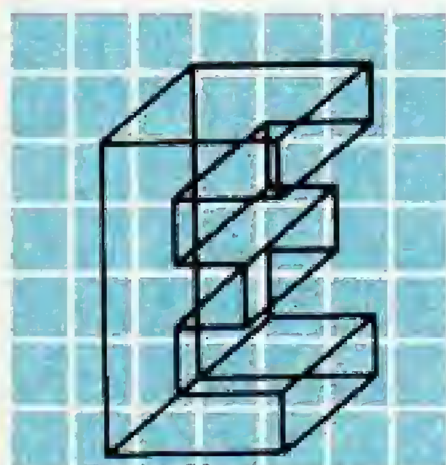
Tabla de equivalencias

Mandato	Sinclair	Amstrad	IBM	Commodore	MSX
— Posición en la pantalla (C = columna, L = línea)	PRINT AT F,C	LOCATE C,F	LOCATE C	—	LOCATE C
— Inicialización de números aleatorios	RANDOMIZE	RANDOMIZE TIME	RANDOMIZE TIMER	—	RND (-1)
— Generación de números aleatorios	RND	RND	RND	RND (1)	RND (1)
— Fin del programa	STOP	END	END	END	END
— Matrices alfanuméricas I, J, K... Indices L = longitud máx.	A$(I,J,..,L)	A$(I,J,K)	A$(I,J,K)	A$(I,J,K)	A$(I,J,K)

Tabla de equivalencias

Mandato	Sinclair	Amstrad	IBM	Commodore	MSX
– Dibujo de un punto en la pantalla	PLOT X,Y	PLOT X,Y,C PSET(X,Y),C	PSET(X,Y),C		
– Dibujo de una línea en la pantalla X1,Y1 = punto inicial X2,Y2 = punto final	PLOT (X1,Y1): DRAW (X2-X1,Y2-Y1)	PLOT (X1,Y1): DRAW (X2,Y2)	LINE (X1,Y1)-(X2,Y2)	–	LINE (X1,Y1)-(X2,Y2)
– Impresión con formato	–	PRINT USING	PRINT USING	–	PRINT USING
– Dibujo de circunferencias X,Y = centro R = radio	CIRCLE X,Y,R	–	CIRCLE (X,Y), R,1,0,360	–	CIRCLE (X,Y), R,1,0,360
– Cambiar de pantalla N = número de pantalla	–	MODE n n = 0 → 2	SCREEN n n = 0 → 3	–	SCREEN n n = 0 → 4
– Borrar pantalla	CLS	CLS	CLS	PRINT CHR$(147)	CLS
– Asignaciones	LET	LET o nada	LET o nada	LET o nada	LET o nada

PEQUEÑA HISTORIA DE LA INFORMATICA



L ordenador supone para el empleado administrativo lo que el bulldozer para el peón. Esta frase fue pronunciada cuando los ordenadores apenas si tenían diez años de existencia, pero el tiempo ha dado la razón a Sir George Thompson, miembro distinguido de la Royal Society. E incluso lo ha dejado corto. Hoy en día los ordenadores invaden nuestra vida cotidiana. Se utilizan para todo: desde serias investigaciones científicas, proyectos de ingeniería, medicina, gestión de los negocios, hasta en tareas que nos resultan más cercanas, como el control de las reservas de billetes de avión, o ferrocarril, o el "control" del niño latoso que queda hipnotizado matando marcianos o arañas peligrosas.

Pero no vamos a hablar al lector de las posibilidades de su ordenador. Este no es el objetivo que nos hemos fijado. Por el contrario, vamos a contarle un cuento ameno (o al menos eso es lo que pretendemos) sobre ordenadores. Vamos a hablar de la jovencísima historia de la Informática.

Dividiremos esta sección en dos partes. Una de ellas será la historia de la informática propiamente dicha, y la otra, algunas anécdotas en las que se vean involucrados los pioneros de esta Ciencia.

Comencemos, pues, en la prehistoria. Como todos sabemos, el hombre comienza a contar utilizando los dedos de sus dos manos. Pero evidentemente, eso no es mucho. Y de ahí pasa a realizar sus primeros cálculos, generalmente ordenando pequeñas piedras en grupos de diez, para más tarde pasar a contar los grupos formados (la palabra cálculo deriva del latín "cálculus" que significa piedra).

Sobre los sistemas de contar, hablaremos detenidamente en próximos fascículos, ya que existen algunos curiosísimos.

Un avance sobre lo anterior lo constituye la primera máquina de contar: EL ABACO. ¿Dónde se "inventó" el ábaco? Es una pregunta difícil, ya que este pequeño instrumento aparece por todo el orbe, desde tiempos inmemoriales, y en civilizaciones absolutamente distintas y distantes, como en la China, en Egipto, Grecia, etc.; pero, ¿sabía el lector que los españoles descubrieron en América otro ábaco muy utilizado en las civilizaciones precolombinas? También hablaremos de ábacos chinos, romanos, egipcios, de la Europa medieval, e incluso japoneses. Pero no se asuste el lector, vamos a hablar además de otras muchas cosas curiosas e interesantes.

Avanzando más aún, el hombre se encuentra con el problema de solucionar la notación adecuada para representar sus cálculos. Es un problema importante. Como ejercicio práctico, podemos intentar multiplicar CXXXVIII por MDCCXLIII. También veremos cómo a lo largo de la historia, o mejor prehistoria, se ha ido resolviendo este problema fundamental, por griegos, fenicios, bizantinos, babilonios, mayas, etc., hasta llegar al perfeccionamiento que le dieron los árabes.

En 1617 aparece otro invento. Son unas pequeñas tablas de cartón, metal, madera o hueso, y su autor es nada menos que Juan Neper, el que inventaría más tarde los logaritmos. Estas tiras, de las que hablaremos más adelante, servían para realizar multiplicaciones complicadas, como, por ejemplo, 3.568 por

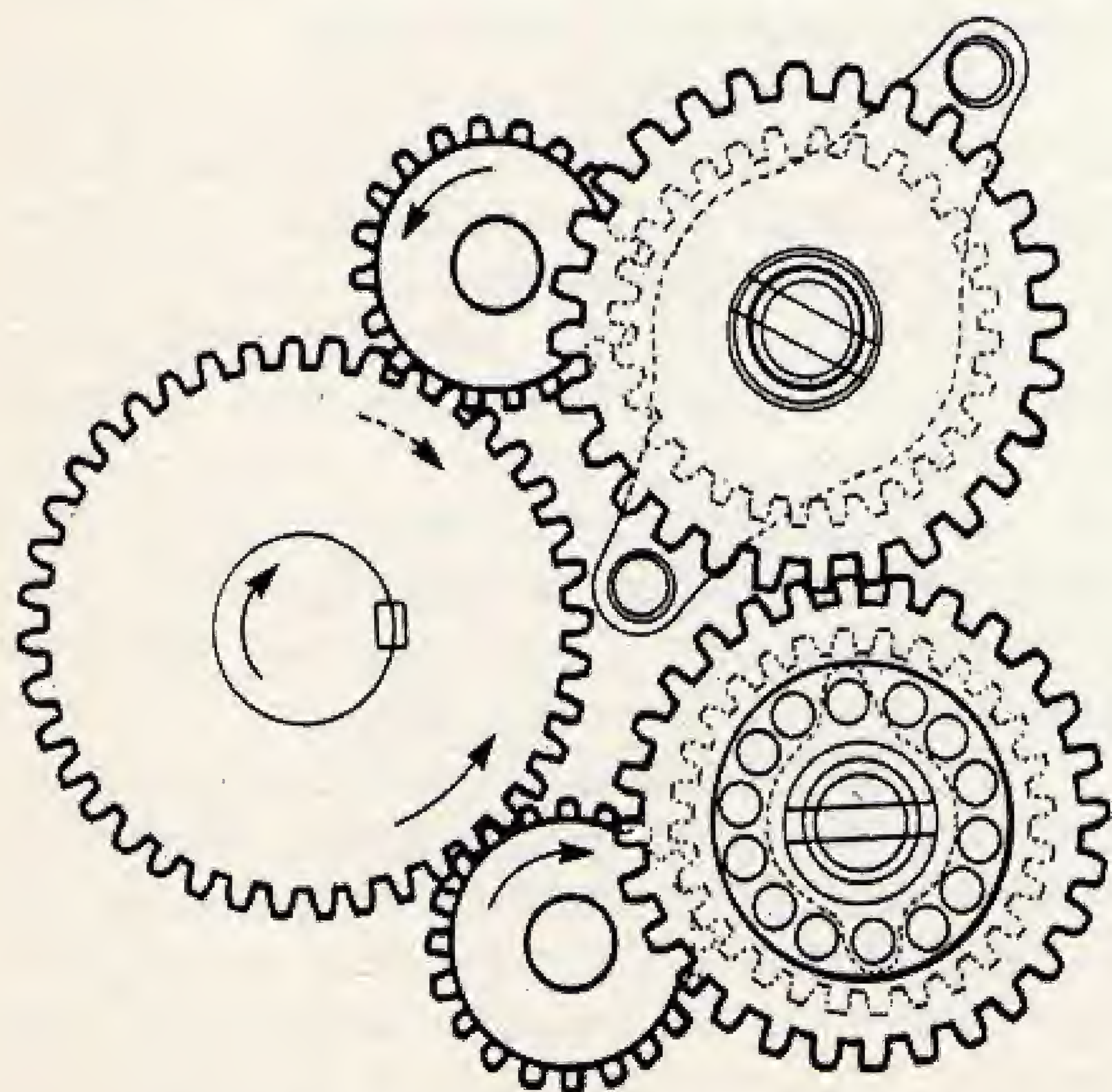
2.845. Pero su invento revolucionario fueron desde luego los logaritmos, que simplificaban enormemente las operaciones de multiplicación y división de números, convirtiéndolas en simples sumas y restas.

Todavía nos queda mucho camino hasta alcanzar el más modesto y sencillo de los ordenadores actuales. Midamos cuidadosamente ese camino ¡con una regla de cálculo!

La regla de cálculo es consecuencia directa del invento de Neper. Los logaritmos de los números se marcan en la regla y las multiplicaciones y divisiones se hacen sumando o restando longitudes. Y ya estamos en el siglo XVII.

Y aquí es donde aparece la primera diferencia de cálculo que se mantiene hasta los ordenadores modernos: unos son digitales, es decir, operan directamente con números y siguen el sistema que tenía el ábaco, que contaba objetos determinados; y otros son analógicos, es decir, utilizan la manipulación de variables físicas análogas a las cantidades que se trata de computar (en un principio se manipulaban ángulos o desplazamientos mecánicos, y después cantidades eléctricas, voltaje, intensidad de corriente, etc.).

Las primeras máquinas de calcular mecánicas son también del siglo XVII. Y, siguiendo el mismo principio que el ábaco, en lugar de contar cuentas, lo que hacen es contar dientes de rueda, o ranuras de una barra lineal. Aquí el problema fundamental lo constituye el arrastre de una posición a la siguiente posición más significativa (en primer lugar, las decenas). Evidentemente, la operación de paso debe ser automática (no como en el ábaco, en el que el usuario realizaba esta operación manualmente). Este problema tampoco es una montaña para nuestros inventores de hace varios siglos: el engranaje se realiza por ejes masivos, con reducción de diez a uno, o mediante engranajes desmontables. Todo ello es tan simple que, una vez explicado, resulta elemental. Lo veremos en los próximos fascículos, así como el sistema de suma de dos números mediante ruedas con trinquete. También veremos cómo restar y registrar los números en un contador, para obtener un resultado en


Diagrama de la máquina analítica de Babbage.

Sabía usted que...

¿Usted ya sabe que un chip o circuito integrado es sencillamente un pequeño rectángulo de silicio en el que mediante técnicas físicas, químicas y fotográficas (fotograbado) se implanta en su interior un circuito electrónico con multitud de transistores microscópicos?

Los primeros circuitos integrados tenían unos 100 componentes por chip, pero las técnicas han ido mejorando hasta el punto de que hoy en día, un circuito integrado puede llevar en su interior hasta un millón de componentes, dispuestos en varias capas.

Como las conexiones son indisociables de los propios componentes de los circuitos, el chip es mucho más fiable y mucho más rápido que los circuitos clásicos. Además, el proceso de miniaturización tiene enormes ventajas prácticas: menor volumen, menor consumo, menor disipación de calor, etc.

Las condiciones de fabricación de los circuitos integrados son muy estrictas. Las impurezas incontroladas pueden estropear los circuitos, y por ello el aire de los talleres se purifica exhaustivamente para eliminar el polvo que lleva en suspensión (en los hospitales, el nivel de polvo admisible es mucho más alto). Concretamente, en el ambiente del recinto de fabricación de los circuitos integrados se exige que el nivel de impurezas sea menor de 2.500 granos de polvo por metro cúbico, mientras que en un hospital, el nivel admisible es de 35.000 granos por metro cúbico.

En general, la fabricación es tan delicada que cuando se lanza la producción de un nuevo modelo, llegan a desecharse hasta el 96 % de los chips fabricados.

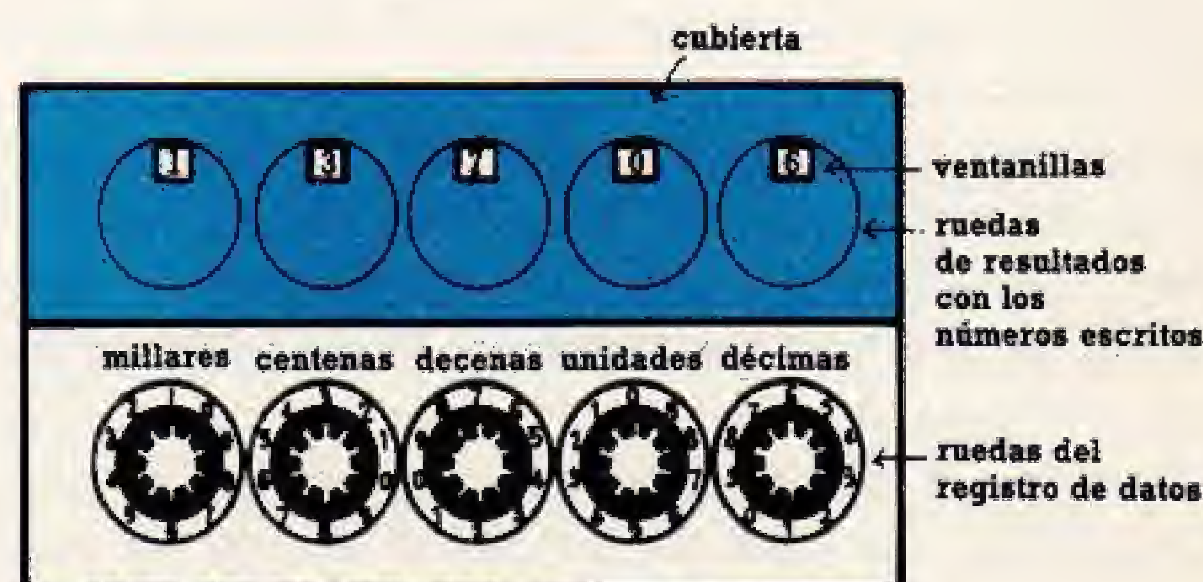
otro contador. Todos estos son problemas ya resueltos con nuestra máquina, la Pascalina, (su inventor fue el matemático y eminente filósofo Blaise Pascal, que la diseñó para ayudar a su padre, que era recaudador fiscal). El proyecto lo realizó a los diecisiete años, pero la primera máquina, de la que quedan algunos modelos en París, sólo apareció en 1642. No nos detengamos más, por el momento en la máquina Pascalina, y sigamos avanzando en el tiempo. Sólo cuatro años después aparece otra máquina, con algunas mejoras sobre la anterior. El proyecto se debe a Leibniz, y en Hannovver pueden verse algunos ejemplares de su máquina, construida hacia 1671. El progreso fundamental de esta nueva máquina frente a la anterior, es que aunque Pascal había conseguido mecanizar las sumas, las multiplicaciones eran lentísimas. Leibniz fue el que logró superar este problema, mecanizando efectivamente la multiplicación. La máquina de Leibniz tenía desde luego innovaciones importantes como por ejemplo la rueda escalonada, de la que hablaremos más extensamente, y hasta hace pocos años se han utilizado este tipo de ruedas de Leibniz en las calculadoras.

Todos estos avances en la automatización de las máquinas se realizaron con grandes esfuerzos. Nuestros inventores eran como nosotros, hombres de carne y hueso, aunque con ciertas cualidades excepcionales. Eran envidiosos con frecuencia, y se desprestigiaban unos a otros. Muchos, también estaban amargados, por el enorme esfuerzo desarrollado, que rara vez era reconocido y apoyado. Menos aún económicamente. La mayoría eran hombres tan innovadores que no podían ser comprendidos por el ciudadano medio de la época.

Las máquinas de las que antes hablamos eran absolutamente artesanales, ya que exigían gran precisión, lo que era imposible obtener con las técnicas de ingeniería de la época. Sin embargo, se hicieron varios intentos de fabricación de máquinas en serie, pero estos proyectos sólo pudieron llevarse a cabo de forma medianamente satisfactoria en el año 1810. En Alsacia, Charles Thomas fabrica unas 1.500 máquinas de este tipo en un período de unos cincuenta años. Algunas de las máquinas llevaban otro sistema diferente del sistema de ruedas de Leibniz. Pero no nos detengamos más por el momento, bastará con que indiquemos que las modificaciones más importantes se debieron a Baldwin y Ohdner.

Hasta el momente hemos hablado de máquinas automáticas, aunque realmente no lo eran. Siempre necesitaban alguien que las manejase. Evidentemente, el avance conseguido en rapidez se veía muy disminuido con la lentitud de las acciones que debía llevar a cabo el operador. Y las miras de nuestros antiguos hombres de ciencia se dirigían a conseguir una máquina *totalmente automática*, es decir, que no requiriera en absoluto la manipulación humana. El padre de esta idea fue Charles Babbage, que casi llegó a crear una máquina que cumpliera esas características. Fue profesor en Cambridge nada menos que en la cátedra del insigne Isaac Newton. ¡La verdad es que apenas si vio a sus alumnos unas cuantas veces, ya que estaba muy ocupado en resolver innumerables problemas: la modernización de la enseñanza de las matemáticas, en Cambridge el observatorio de Greenwich, la Oficina Postal, la industria tipográfica del momento, y muchos otros organismos. La industria de alfileres del Reino Unido también le debe gratitud! ¡Pero esos pequeños trabajos los realizaba en los días laborables! Los domingos tenía todavía tiempo para conducir un tren especial, y llevar a cabo una serie de medidas dinamométricas para los ferrocarriles.

Su aportación a la mecanización de cálculos fue enorme. Su primera máquina, la *"Máquina de Diferencias"*, fue concebida para calcular e imprimir tablas de funciones matemáticas. Con ella era capaz de representar cualquier función mediante polinomios con un grado de exactitud suficiente. En 1822 realizó una demostración de la máquina y obtuvo una ayuda financiera considerable del gobierno, y de la Royal Society para construir una máquina mayor. Pero Babbage era un científico, no un empresario, y aunque el gobierno le entregó sumas enormes para la fabricación de la máquina, la verdad es que no, Babbage no



Esquema de la máquina de calcular de Pascal.

estaba contento con su máquina de diferencias, su estudio ya no le interesaba, y lo que deseaba era diseñar otra máquina absolutamente automática, que partiera de concepciones completamente distintas. No deseaba perder el tiempo en sistemas anticuados... que se siguen repitiendo en el siglo XX. Efectivamente, en nuestros días se han utilizado esquemas de la máquina de diferencias en ciertas máquinas contables.

Pero la idea que Babbage llevaba en su cabeza era muchísimo más ambiciosa: su máquina debía ser capaz de realizar cualquier cálculo automáticamente, debía ser una máquina universal, capaz de realizar cualquier tarea si era programada (instruida) adecuadamente —en contraposición con las máquinas de las que antes hablamos, que realizaban sólo una tarea concreta, para la que habían sido pensadas—.

Babbage pensó en dotar a su máquina de los siguientes elementos (precursores de nuestros ordenadores de hoy):

a) Un lugar de **almacenamiento** (la memoria actual) donde se guardaran los números del problema a resolver y también los generados por las propias operaciones de resolución del problema.

b) Una unidad capaz de realizar las operaciones aritméticas necesarias, utilizando los números almacenados.

c) Un sistema de control, para que la máquina realice las tareas siguiendo la adecuada secuencia de operaciones.

d) Un medio para introducir a la máquina los datos (números e instrucciones para que realice las operaciones en el orden correcto).

e) Otro sistema de salida que muestre los resultados de una determinada tarea.

Como puede ver el lector, la concepción de la máquina de Babbage es la misma que la de nuestros rápidos y eficaces ordenadores modernos, y, sin embargo, cuántos años han sido necesarios para llegar a los ordenadores de hoy. Babbage fue un hombre precursor de su tiempo, aunque tuvo enormes dificultades en sus investigaciones, ya que debemos recordar que los grandes inventores deben además luchar para que sus ideas se lleven a la práctica, se realicen. Todo lo plasmado en el papel, debe cobrar vida, ser un objeto real. Y los problemas son enormes. Los materiales de fabricación de la época no son todo lo fiables que se necesita. Y con frecuencia nuestros investigadores recurrían a otros inventos, que difícilmente se pueden relacionar con lo que nos ocupa. Veamos, por ejemplo, el sistema de introducción de los datos en la máquina. Fue una réplica del utilizado en Francia por Jacquard en sus telares. A su vez Jacquard había basado su sistema en otro anterior, modificándolo adaptándolo a sus necesidades concretas, incluso mejorándolo. El sistema de Jacquard, del que desde luego hablaremos más extensamente, fabricaba jerseys y telas con los dibujos más complicados, utilizando combinaciones de colores increíbles. Para combinar los hilos utilizaba varillas metálicas y fichas perforadas, que indicaban a la máquino cuándo debía introducirse una determinada varilla con ciertos hilos de colores y cuándo no (naturalmente trabajaba con gran número de varillas). Babbage admiraba tanto a Jacquard que tenía un retrato suyo (creado con el sistema anterior) en el salón de su casa. Era la admiración de todo el que lo veía por primera vez, ya que nadie observaba que estaba confeccionado en un telar. Gracias a Babbage, Jacquard y otros como ellos, la industria del

Sabía usted que...

¿Sabía usted que Thomas Watson Senior, fundador de IBM, fue carnicero y vendedor de pianos?

Desde luego, su vida siempre estuvo ligada a las ventas. Había nacido en 1874 en el seno de una familia oriunda de Escocia. En 1892 comenzó su trayectoria profesional como vendedor de equipos mecánicos (pianos, máquinas de coser, registradoras, etc.). Como tal corredor de comercio, ya conseguía comisiones récord (1.225 dólares semanales, cuando sus compañeros no conseguían alcanzar los 200).

Su actividad comienza ya a tener un cierto paralelismo con lo que sería unos años después, cuando es contratado por la NCR, a cuya cabeza se encuentra otro hombre insigne: Thomas Patterson. De su nuevo patrón, Watson aprende cómo debe gobernarse una empresa. Con él también asimila cómo deben ser las relaciones con sus empleados, a los que mima en sus necesidades, pero de los que exige una disponibilidad que ha sido artífice principal del éxito de la Compañía.

Desde la fundación de IBM en 1924, hasta su muerte en 1956, su vida estuvo tan intrínsecamente ligada a la Compañía que es difícil separar uno de la otra.

metal experimentó considerables avances. Pero ya hablaremos de las fichas perforadas, su historia y su decadencia.

Babbage trabajó en su **máquina analítica** hasta su muerte, en 1871. Dio conferencias en el continente, pero murió amargado e incomprendido. Sus teorías eran demasiado modernas y ambiciosas, y además, y quizá por ello, su máquina nunca llegó a terminarse.

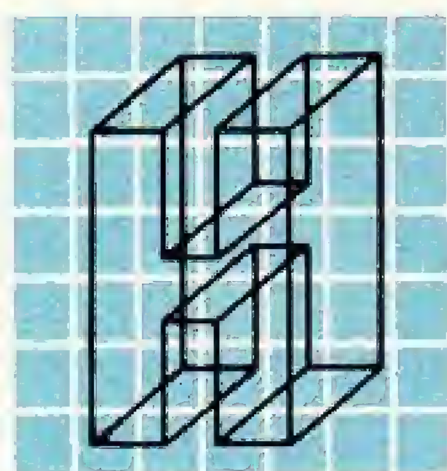
Pero ¿dónde aparece una mano o mejor un cerebro femenino en nuestra historia? Evidentemente, en el siglo pasado no muchas mujeres tenían acceso a buenos libros, e incluso a disfrutar de la compañía de hombres excepcionales. Por todo lo anterior, resulta evidente que la primera mujer interesada en la mecanización tenía que ser noble: Lady Lovelace, hija única de Lord y Lady Byron. (Menciono a Lady Byron porque es de Lady Byron de donde heredó Lady Lovelace su afición a las matemáticas.) Esta inteligente dama siguió los pasos de Babbage a través de algunos artículos publicados sobre él, tradujo y recopiló sus artículos y conferencias, y llegó a comprender su proyecto totalmente. Es más, exponía las teorías de su maestro con mucha mayor claridad y fluidez que él mismo, y podemos constatarlo en documentos de la época. A ella se atribuye la frase acertadísima "La máquina analítica tiene ninguna pretensión de originar nada, sólo es capaz de hacer aquello que sepamos indicarle que haga".

Sabía usted que...

¿Sabe que uno de los primeros ordenadores, el ENIAC, funcionaba con válvulas y tenía tal consumo eléctrico que cada vez que se conectaba a la red bajaba la tensión en la pequeña ciudad próxima a Filadelfia donde se encontraba?

El ENIAC pesaba 30 toneladas, constaba de un millón de piezas y utilizaba 19.000 válvulas. Su principal inconveniente era que el paso de una determinada tarea a otra resultaba muy complicado (ya que requería manipulaciones en el cableado de la máquina), y la memoria era muy pequeña. Su mejor cualidad era la enorme rapidez en los cálculos (frente a los métodos y máquinas existentes en la época). Fue diseñado para resolver problemas militares (cálculo de trayectorias, etc.).

Visión artificial (I)

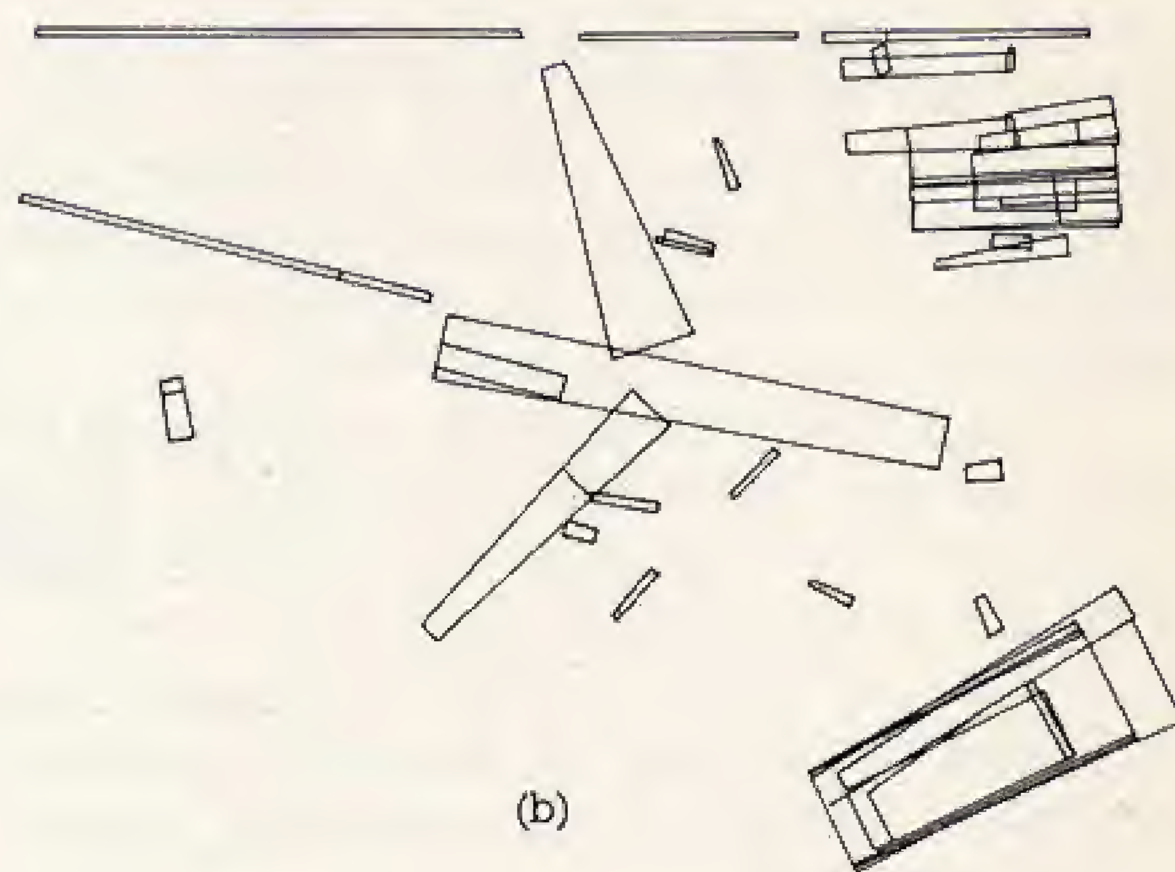


HACE más de veinticinco años que los investigadores comenzaron a preocuparse de la visión artificial, es decir, de dotar de "ojos" a las complicadas y serviciales máquinas que iban creando. En un principio, se pensó que la tarea era sencilla, pero el tiempo transcurrido desde entonces ha demostrado que la tarea no es en absoluto sencilla. Sin embargo, la visión por ordenador ha experimentado un avance notable en estos últimos años, ya que hoy en día, además de utilizarse las técnicas clásicas de Proceso de las Imágenes y Reconocimiento de Formas, el desarrollo de la Inteligencia Artificial ha supuesto una ayuda enorme en la resolución del problema.

Realmente, ¿cómo podríamos definir la visión? La visión, según Marr, es "un proceso que parte de imágenes pertenecientes al mundo externo, y de ahí produce una descripción útil al que mira, liberándole de aquella información que pudiera ser irrelevante". Simplificando la definición anterior, "la visión es un proceso que parte de imágenes pertenecientes al mundo externo, las procesa, y de ahí obtiene una descripción útil, eliminando toda información que pudiera ser superflua.



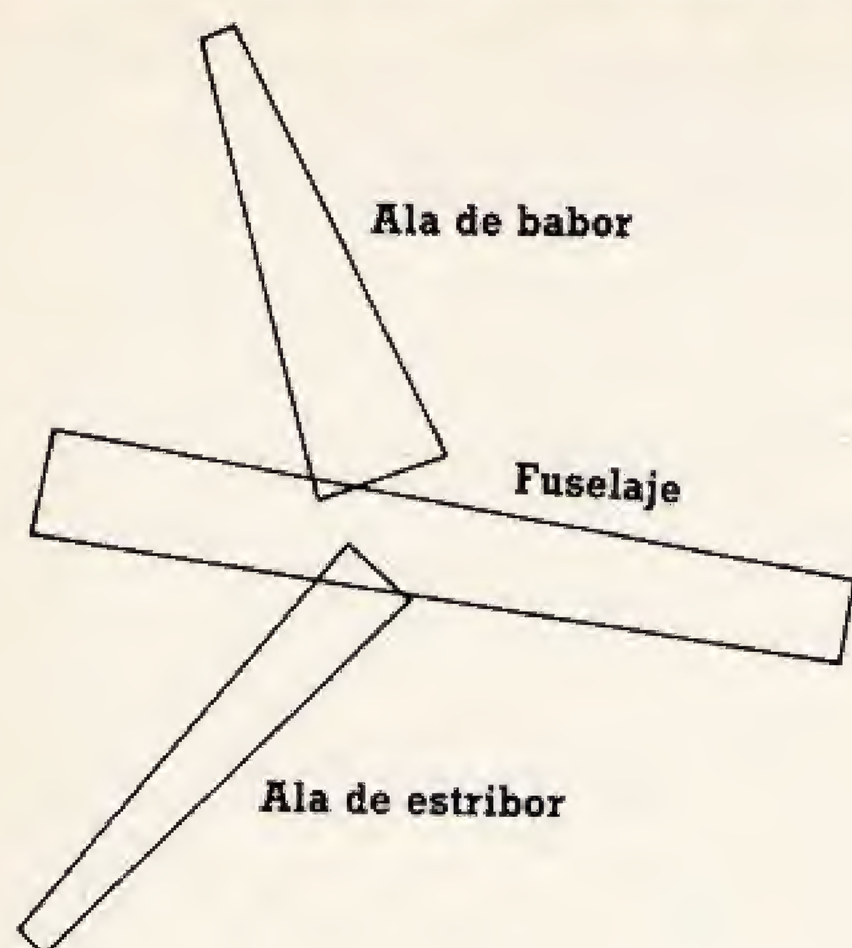
(a)



(b)

A pesar de que el investigador ha cargado las tintas en la tarea de simplificar la información que obtiene de las imágenes, ciertas acciones fisiológicas del cerebro que parecen a simple vista elementales, resultan enormemente complicadas para un ordenador capaz de resolver operaciones complejas en milésimas de segundo. La enorme cantidad de datos recogidos para ser procesados hace que el tiempo de proceso sea largo, complicando enormemente la tarea.

En concreto, ¿cómo se aborda el problema? Se trata de un proceso secuencial, por el cual partiendo de la propia imagen, se extrae información sobre sus características (bordes, elementos notables, texturas, etc.)



para a continuación pasar a codificarlas adecuadamente (se analizan y agrupan en representaciones simbólicas, como zonas, contornos, etc). Con toda la información anterior, y tomando unos modelos previamente almacenados en el ordenador, se obtiene una interpretación semántica de la escena.

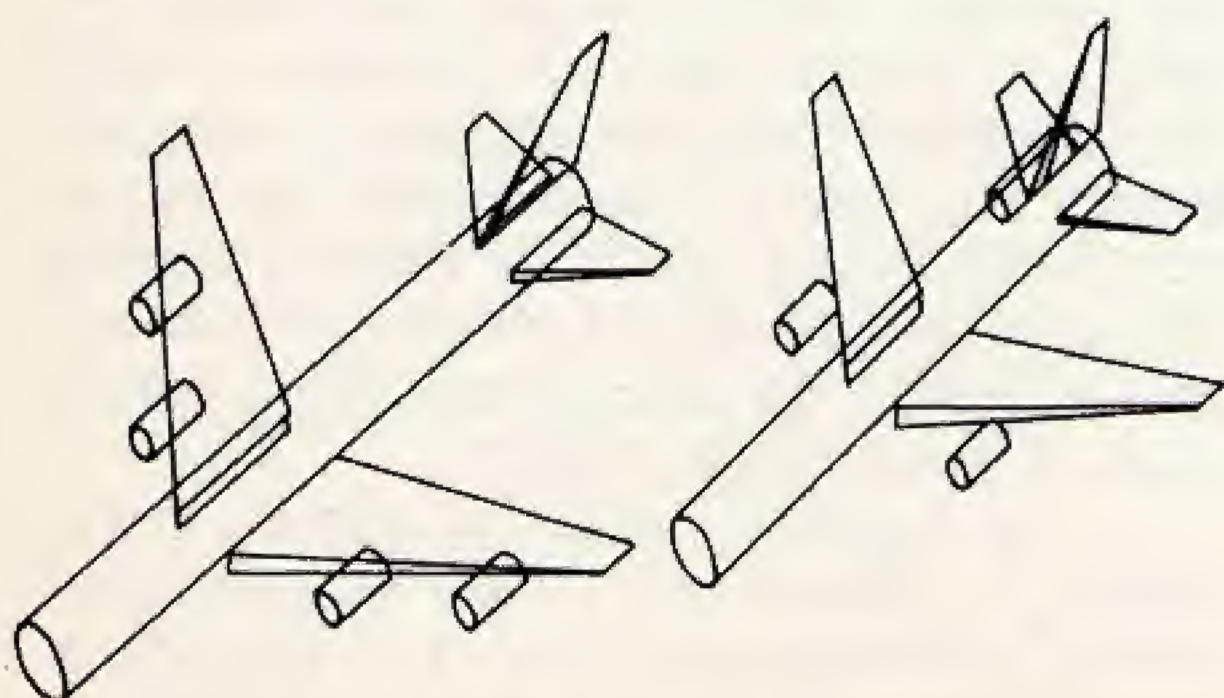
Otra forma de abordar el problema es partir de una "hipótesis-verificación" e irnos guiando por descripciones de la escena generadas por conocimientos previos.

Pero, centrándonos en el primer método, que es el más usual en las aplicaciones comerciales, analicemos los elementos necesarios para trabajar.

En primer lugar, necesitaremos una cámara de vídeo, que es el sistema más utilizado para tomar la informacion sobre la imagen.

Fijémonos en el ojo humano: para captar las imágenes, el ojo dispone de un sensor óptico, la retina. El emulador artificial de este sensor humano es precisamente la cámara de vídeo. El papel que realiza la cámara de vídeo se reduce a convertir en señales eléctricas las informaciones que ha tomado de la imagen.

Sin embargo, la señal que devuelve la cámara de vídeo es una señal analógica, y como deseamos que el ordenador la procese, será necesario digitalizarla.



Modelos del Boeing-747 y del Lockeed L-1011.

Para digitalizar la imagen y convertirla en un conjunto de códigos (números binarios) que pueden ser procesados por un ordenador, se utiliza un conversor analógico/digital.

La información digitalizada suele almacenarse frecuentemente en forma matricial en una memoria intermedia. Suele llamarse "cuadro".

A continuación, la información sobre la imagen (cuyas características, parámetros, bordes, están ya codificados), es procesada por uno o más microprocesadores (generalmente muchos).

Por último, cuando la visión artificial se aplica a un proceso específico (principalmente en robótica), la señal digital es convertida a analógica por un convertidor, para que pueda ser utilizada por el sistema de control de la máquina de que se trate.

Por tanto, resumiendo, el sistema de visión artificial consta, básicamente, de: una cámara para la captación de imágenes, con su controlador correspondiente, un conversor analógico-digital (que suele incluir elementos para el preprocesamiento de la imagen y su almacenamiento) y un sistema de reconocimiento de formas a partir de la imagen digitalizada, para su posterior clasificación.

Veamos ahora los tipos de cámara que podemos utilizar. Fundamentalmente, existen dos tipos distintos: la cámara vidicón, y la de estado sólido.

Las cámaras vidicón, "inspeccionan" la superficie sobre la que se proyecta la imagen línea por línea, generando por cada punto o pixel una señal (tensión), que es proporcional a la intensidad luminosa que tiene dicho punto. El número de líneas depende de la resolución de la cámara, y las más utilizadas son de 100, 256 ó 512 líneas. Estas líneas están divididas a su vez en 100, 256 ó 512 puntos, o elementos gráficos, a los que se denomina corrientemente pixels (contracción de "picture elements" o "elementos de imagen").

En las cámaras de estado sólido, se considera una retícula, en la que cada pixel es a su vez considerado individualmente como sensible a la luz. Así, pues, el sensor de este tipo de cámara estará formado por una matriz de elementos fotosensibles.

La señal que producen ambos tipos de cámaras es, como antes explicamos, en los dos casos una señal analógica. Como lo que desea-

mos obtener es una señal digitalizada, es decir, un conjunto de valores discretos (cada uno de los cuales representa un nivel de gris), utilizaremos un conversor analógico/digital.

Veamos cómo se transforma una imagen en información digital.

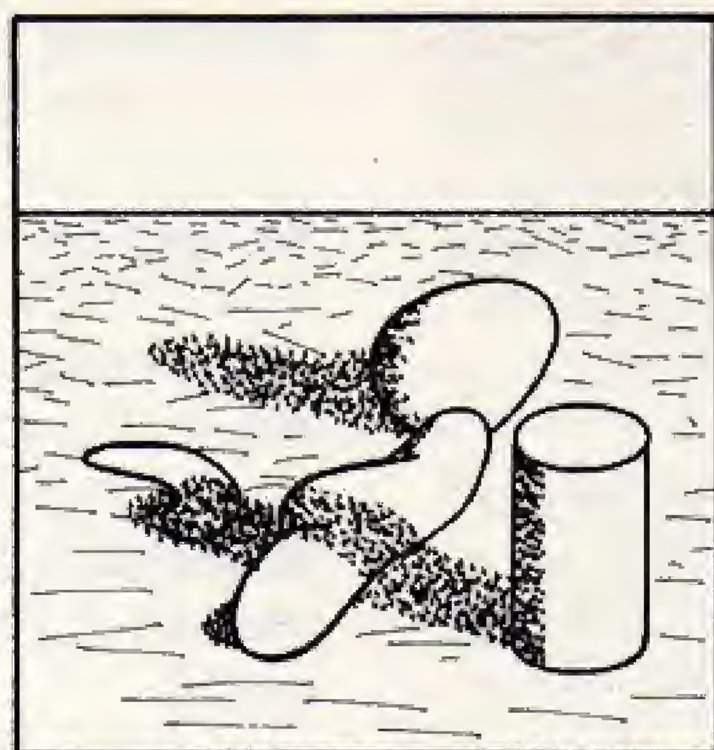
La imagen es captada por un sensor (dispositivo sensible a la luz).

Sobre la imagen que ha captado el sensor se "coloca" una retícula que divide a la imagen en pequeñísimos puntos, y se mide la luminosidad de cada uno de esos puntos.

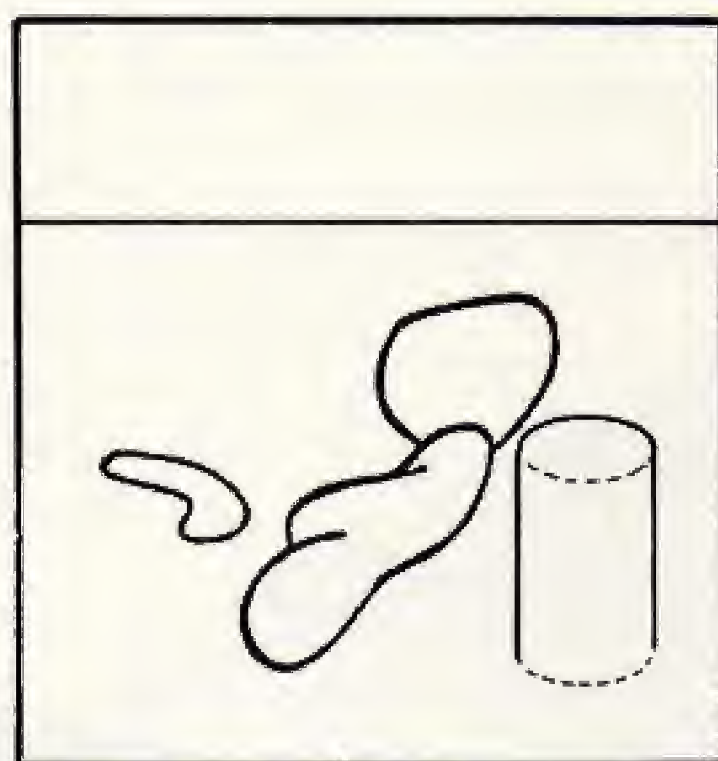
Con los valores de la luminosidad de cada uno de los puntos se crea una matriz, en la que los distintos valores de luminosidad, que son números finitos, indican las distintas tonalidades de gris.

Naturalmente, a mayor resolución de la imagen, mayor número de cuadradillos o celdas de las que se toma la información, más niveles de gris y mayor calidad y definición.

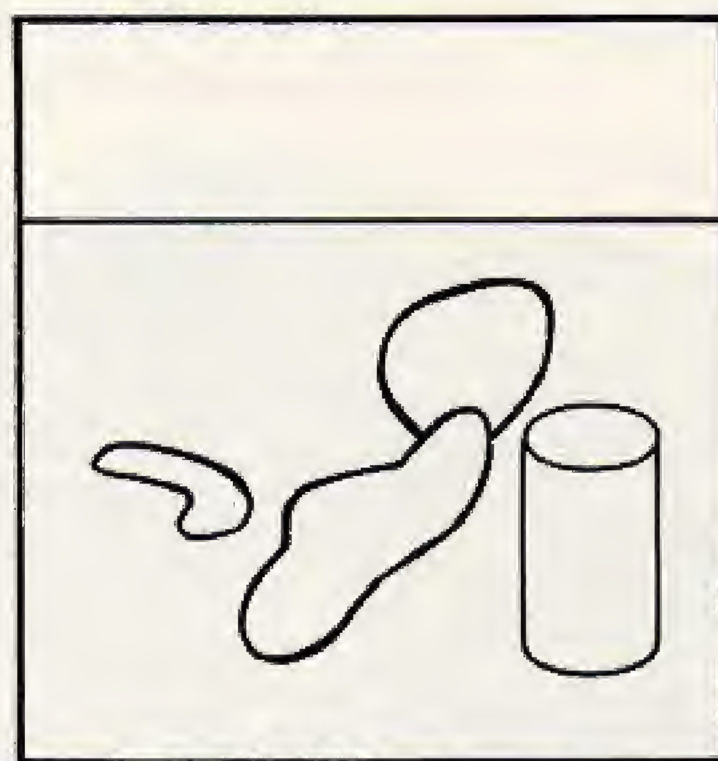
Si deseamos además obtener imágenes artificiales en color, todo lo anteriormente explicado es válido, pero en este caso la toma de información se realiza utilizando tres cámaras (rojo, verde y azul), o una única cámara que disponga del filtro adecuado. En cualquier caso, si trabajamos con el color, dispondremos



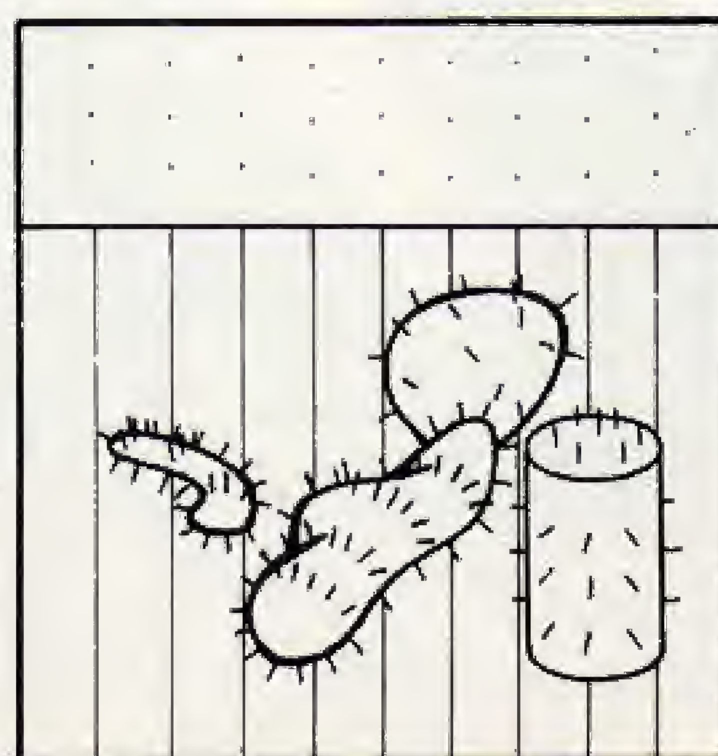
Escena original.



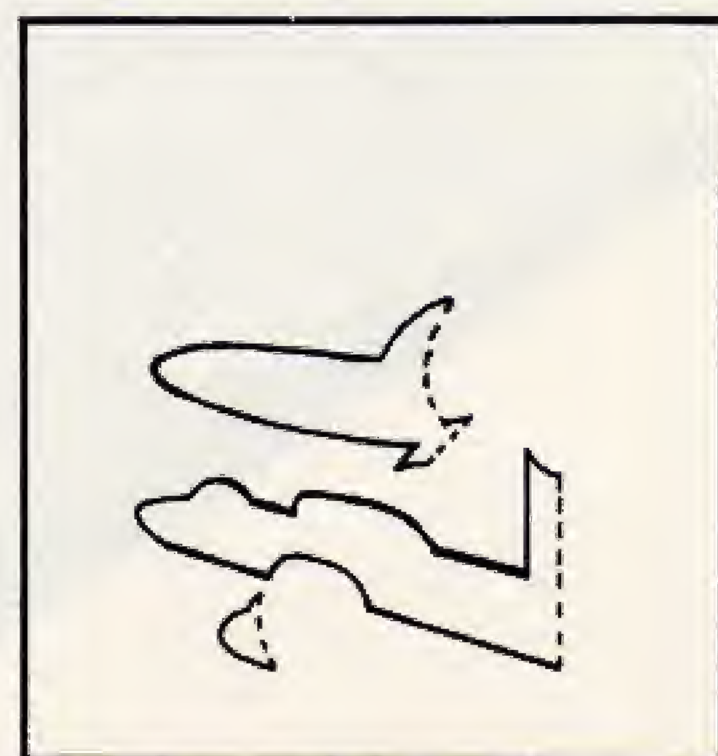
Distancia.



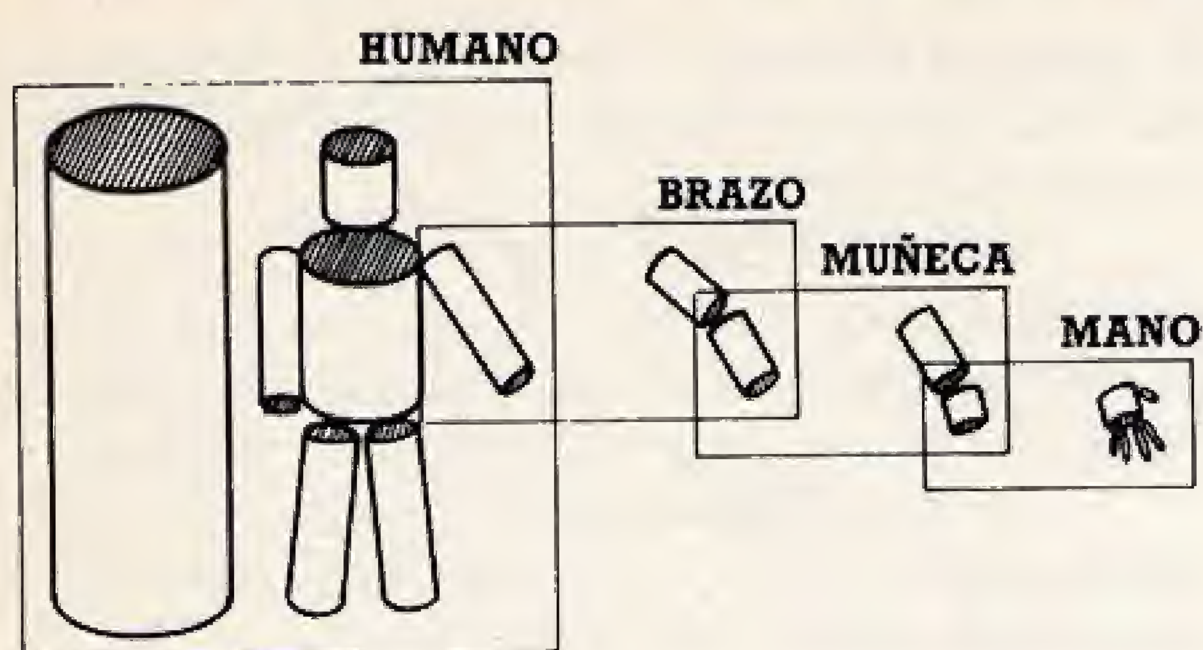
Reflectancia.



Orientación (Vector de).



Iluminación.



Detalle jerárquico de un modelo humano tridimensional.

de tres valores (rojo, verde y azul) cada punto o pixel correspondiente a la imagen.

A partir de este momento ya podemos procesar la señal, que tendrá tantos niveles de gris como distintas combinaciones se puedan hacer con los bits de que se componga cada número, es decir, si el conversor A/D es de 8 bits, existirán 2 elevado a 8, es decir, 256 niveles de gris.

El ordenador utilizará la señal digitalizada para deducir aquellas características de la imagen que necesite cada aplicación en particular. Para la deducción de estas características empleará algoritmos matemáticos específicos. Unos de los más utilizados son los obtenidos en el Stanford Research Institute, por lo que se conocen con el nombre de SRI.

Para la aplicación de estos algoritmos es muy importante, antes de aplicarlos, simplificar al máximo la información obtenida. Un buen sistema es convertir la imagen (que tenía muchos niveles de gris) en imagen binaria. Para pasar una imagen con varios niveles de gris a imagen binaria, se establece un determinado umbral de intensidad, de forma que cada pixel de la imagen que lo supera toma un valor 1, y cada pixel que no lo alcanza, toma un valor cero. Así, pues, habremos obtenido una imagen con menos información, pero por ello mucho más útil para aplicarle los algoritmos, que nos permitirán deducir su perímetro, área y muchas otras características.

Los principales investigadores en la construcción de algoritmos SRI fueron R. O. Duda y Gerard Agin.

Nada más plantearse el problema resultó evidente que sería imposible diseñar algoritmos que se ejecutaran en tiempo real y sirvieran para imágenes de tipo general si no se simplificaban tremendamente las condiciones de aplicación y las características a examinar.

Simplificando nosotros también, indicaremos que las condiciones que debe cumplir



una imagen para poder ser analizada por estos algoritmos son las siguientes:

a) Debe ser una imagen binaria.

b) Las partes que forman el objeto sujeto a reconocimiento no deben tocarse ni superponerse. Si fuera así, se considerarían como una única parte.

c) La imagen debe ser estable. No puede cambiar de posición.

Con este tipo de restricciones, los algoritmos SRI, sin embargo, son muy útiles, y en la práctica, existen numerosísimas aplicaciones industriales en las que satisfacen las necesidades.

Los algoritmos SRI pueden determinar casi 50 características diferentes del objeto (perímetro, área, momento de inercia, radio máximo, etc.).

La visión artificial ha experimentado un enorme avance en los últimos años, debido principalmente al enorme auge de los microprocesadores de 16 y 32 bits. Además, tanto procesadores como cámaras de vídeo han sufrido un abaratamiento considerable, lo que ha ayudado a que la industria demande este tipo de sistemas, que, como veremos, son utilísimos, y pueden desarrollar multitud de tareas.

Actualmente son enormes las posibilidades que ofrece la Visión Artificial:

A) Automatización de infinidad de procesos industriales:

Tecnología VLSI, soldaduras de terminales, organización de chips.

Guiado automático de todo tipo de herramientas (corte, soldaduras, plantas de embotellamiento, etc.).

Brazos automáticos: empaquetado y clasificación de objetos, etc.

B) Proceso de imágenes obtenidas vía satélite.

Previsión meteorológica.

Creación de Cartas y Mapas.

Estudio de zonas vírgenes sin explorar, por ejemplo, para un mejor aprovechamiento de sus recursos, agua, bosques, etc.

C) Localización de fallos y errores de producción.

En fundiciones, impurezas y fracturas.

En empresas de vidrio, detección de posibles burbujas o fallas en el material.

Detección de posibles fallos, cortocircuitos, etc., en circuitos impresos, y otro material electrónico, etc.

D) En Medicina, Biología, Bromatología, etcétera.

Detección de ciertas características especiales a partir de imágenes, para diagnósticos (rayos X, ecografías, etc.).

Clasificación y recuento de células o tejidos de todo tipo.

Medición de parámetros de órganos, etcétera.

E) En aplicaciones militares la visión artificial puede tener una importancia enorme; veamos por ejemplo:

Seguimiento y localización de objetos en movimiento.

Trayectorias automáticas (misiles, etc.).

Además, en todos los campos anteriores, mejorando la tecnología, al favorecer la entrada de datos gráficos al ordenador.

TERMINOLOGIA

GLOSARIO DE TERMINOS DE VISION ARTIFICIAL

Algoritmo para la visión. Es un conjunto de operaciones desarrolladas en una secuencia preestablecida y conducente a la resolución de un problema específico que se repite frecuentemente.
Tanto en el área de preprocesamiento de imágenes, como en el posterior reconocimiento de formas, se utilizan numerosos logaritmos: algoritmos de la pirámide, o del árbol analítico de la cuadrícula, para el reconocimiento de bordes, establecimiento de cilindros generalizados para la identificación de objetos dentro de una escena, técnicas de coincidencia con una plantilla o métodos lingüísticos para clasificación de patrones, etc. La mayor parte de estos algoritmos están incluidos en los propios equipos de visión, o en los de control del sistema.

Controlador. Convierte la imagen obtenida por la cámara en forma analógica en una imagen digital equivalente. El controlador establece la velocidad de extracción del sensor y produce la información auxiliar que sea necesaria para el proceso. Generalmente va provisto de un dispositivo muestreador, y de un convertidor A/D para cuantificar y digitalizar respectivamente la información visual.

Estructuras en Array Procesor. Estructuras formadas por un conjunto de procesadores muy simples (en algunos casos de 1 solo bit) que realizan tareas aritméticas y lógicas muy simples (en algunos casos de un solo bit) tienen que estar siempre conectados a un ordenador central, que es el que realiza la carga de los programas en la memoria de la Unidad Central, o en las memorias locales de los procesadores del array.
De lo anterior se deduce que un Array Procesor no es nunca un dispositivo autónomo, y se utiliza para el proceso en paralelo de algunos algoritmos que sean susceptibles de adaptarse a su estructura.
Normalmente, las filas (arrays) de procesadores se conectan entre sí en estructuras complejas (mallas) que pueden llegar a tener hasta 128 × 128 procesadores elementales, como sucede en el computador STARAN.

Extracción de las características de un objeto. En un primer reconocimiento del objeto, y partiendo de una representación simbólica, por ejemplo, de sus bordes se extraen las características que lo definen.
Estas características pueden ser muy variadas y dependen del equipo utilizado y del objetivo a conseguir. Los algoritmos SRI definen casi 50 características distintas de un objeto: área, perímetro, momentos de inercia, relación del perímetro al área, centros de gravedad, distancias de ciertos elementos notables al centro de gravedad.

Histograma. Gráfico en el que aparecen aquellos pixels que tienen cierta intensidad luminosa establecida. De las imágenes de gradiente se pueden obtener histogramas sin excesivas dificultades, y seleccionando el umbral adecuado, se puede pasar a imágenes binarias.

Imagen de gradiente. En una imagen dada, se calcula el gradiente de cada uno de los pixels con respecto a los pixels adyacentes. De este modo, se crea una nueva imagen de gradiente que pone de relieve la diferencia de intensidad entre las líneas de los bordes y el resto de la imagen. Si no existe gradiente de intensidad entre un pixel y los pixels

adyacentes a dicho pixel, se le asignará un valor cero. Con esta técnica todas las zonas homogéneas, claras y oscuras toman un valor 0, quedando evidenciados los bordes (zonas en las que los valores son distintos de cero).

Multiprocesador. Conjunto de procesadores, cada uno de los cuales dispone de su propia Unidad de Control, pero que comparten algún dispositivo común. Su conjunto constituye una estructura de control más compleja.

Nivel de gris. Se dice que una imagen tiene un determinado nivel de gris cuando está formada por puntos de distinta intensidad luminosa, que pueden tomar más de dos valores diferentes. Si la intensidad sólo puede tomar dos valores, la imagen estará en blanco y negro.

Procesamiento de imágenes. Conjunto de tareas previas a realizar en cualquier sistema de visión artificial para mejorar la calidad de la imagen y preparar las diferentes zonas de ella para su posterior análisis.
Incluye dos grupos diferenciados de tarea: por un lado, las técnicas de reconstrucción de la «imagen ideal», y por otro las de mejora de la calidad de la imagen de entrada.
En el primer grupo se utilizan técnicas como la corrección geométrica, la modificación de la escala de gris y los procesamientos para obtener ángulos o suavizar curvas, análisis de regiones, etc.
En el segundo grupo hay que realizar tareas como la eliminación del ruido, o el subrayado de alguna de las características básicas de la imagen.

Reconocimiento de formas. Tareas a realizar en un sistema de visión para el análisis de los objetos de la imagen y su identificación. Suelen seguir al preprocesamiento de las imágenes. En las tareas de reconocimiento de formas se suelen considerar dos etapas: extracción de características de la imagen e identificación. Las características que se suelen considerar dentro del escenario observado son: imágenes intrínsecas de los objetos (distancia, reflectancia, orientación, iluminación, luminosidad, especularidad, etc.), regiones, separación de regiones y amplificación por campos, textura, etc.

Segmentación. Es el proceso por el cual se seleccionan los objetos de interés dentro de la imagen, con el fin de simplificarla. Generalmente se utiliza la técnica de detección del borde, que suele aplicarse a imágenes binarias. Para obtener una imagen binaria de una imagen de nivel gris, basta con seleccionar el **umbral** adecuado en el histograma de la imagen. Una vez establecido el umbral U, se asigna a todos los pixels cuya intensidad sea superior a U un valor 1, y a aquellos cuya intensidads sea inferior a U, un valor cero.

Vector de características. Conjunto de características representativas de un objeto que se utilizan para representarlo y para su posterior análisis o identificación.

VOCABULARIO DE INFORMATICA

Abaco (abacus). Dispositivo de origen muy antiguo que sirve para contar y realizar pequeñas operaciones elementales de suma y resta. Existen muchos tipos, pero generalmente consiste en unas cuentas que van insertadas en unas guías. El operador desliza estas cuentas o discos por las guías para realizar las operaciones.

Abortar (abort). Terminación forzada de un programa o procedimiento. Puede ser debido al sistema operativo, cuando se fuerzan condiciones no aceptables por el sistema o a un acto voluntario por parte del operador.

Abrir un canal. Conexión lógica de la memoria central del ordenador con un determinado periférico. Terminada la operación, el canal debe **cerrarse,** para asegurarnos de que la transferencia de datos se ha completado satisfactoriamente.

Abscisa. Una de las coordenadas cartesianas. Normalmente se representa en el eje horizontal y se marca con una x.

Absoluta, dirección. Dirección fija de la memoria del ordenador, que no depende por tanto de la dirección origen de un módulo de programa (en contraposición a la dirección relativa).

Absoluta, comparación. Comparación independiente del signo del número que se compara.

Absolute code (ver código absoluto).

Acceso aleatorio (random access). Posibilidad de acceder directamente a un elemento cualquiera de un fichero o de una memoria, sin el paso intermedio por un índice, ni ninguna otra secuencia de acceso.

Acceso secuencial. Acceso a los datos de forma que la lectura de los bloques de datos o registros se realiza en el orden físico en el que aparecen, hasta alcanzar el dato deseado.

Acceso en paralelo. Acceso a un dispositivo de almacenamiento por el que se transfieren varios bits de información simultáneamente (en contraposición con el acceso en serie).

Acceso en serie. Acceso a un dispositivo mediante la transferencia sucesiva (bit a bit) de los datos.

Acceso, tiempo de. Intervalo de tiempo entre el instante en que se solicita un dato de un dispositivo de almacenamiento y el momento en que el dato está disponible en la CPU.

ACK. Siglas para identificación del carácter de acknowledge.

Acknowledge, carácter de. Carácter de «recibido». Este carácter es emitido por un dispositivo receptor de información como respuesta a una llamada o a un envío de datos.

Acoplado (coupled). Generalmente se refiere a dos sistemas que pueden ser independientes, y están funcionando con algún tipo de cooperación.

Acoplador acústico. Dispositivo de interfaz para la conexión de un periférico o un terminal a la red telefónica a través de un teléfono convencional (no un equipo preparado especialmente para la transmisión de datos). La conexión se establece marcando un número telefónico del mismo modo que se realiza para una conversación normal. Posteriormente, se acoplan auricular y micrófono del teléfono en unos receptáculos preparados al efecto en el acoplador acústico.

Acoplamiento mutuo (interface). (Véase interfaz).

Acoustic coupler. (Véase acoplador acústico).

Acoustic delay line. (Ver línea acústica de retardo).

Actual, valor. Valor que tiene una variable o un parámetro en el momento en el que se considera.

Acumulador. Registro en el que se almacena temporalmente el resultado de una operación aritmética o lógica. En ocasiones, se utiliza una posición convencional de memoria (no un registro) a modo de acumulador.

ADA. Lenguaje de Programación de alto nivel, desarrollado para el Departamento de Defensa de los EE.UU. y dotado de características especiales para trabajar en entornos de multiproceso en tiempo real, y para el manejo de gráficos. Se le dio ese nombre en honor de Lady Ada Lovelace, considerada la primera programadora del mundo.

ADC, Analog/Digital Converter. (Ver analógico/digital, conversor).

Address, constant (format, part, register). (Ver constante —formato, parte registro— de dirección).

ADP, Automatic Data Processing. (Ver proceso de datos automático).

Agrupamiento. Reunión de varios registros para formar un bloque que puede transmitirse en una sola operación. Se suele dotar al bloque de una cabecera independiente de la de los registros que lo forman.

Ajuste (justificación). También llamado justificación. Proceso de alineamiento de un dato o un texto en el espacio para él reservado: encuadre de un texto entre sus márgenes en un procesador de textos, aproximación de un carácter (o número) a la derecha (o a la izquierda) dentro del campo en que se almacenan, etc.

Aleatorio, acceso. (Ver acceso aleatorio).

Aleatorio, número. Número obtenido al azar, es decir, sin la utilización de ninguna secuencia o algoritmo predefinido para obtenerlo. Normalmente, en los ordenadores se aceptan como aleatorios los números que suministran las funciones correspondientes (en BASIC, RND, por ejemplo), aunque realmente estos números son sólo seudo-aleatorios.

Alfanumérico. Carácter que puede ser un número o una letra alfabética.

Algebra de Boole. Estructura algebraica definida sobre un conjunto mediante dos operaciones que cumplen unas determinadas propiedades. El conjunto (0,1), cuyos valores pueden ser tomados por los bits de un ordenador digital, con las operaciones Y (AND) y O (OR) tiene estructura de álgebra de Boole.

ALGOL. Lenguaje de alto nivel concebido básicamente para la programación de procesos matemáticos. Del lenguaje ALGOL ha derivado el popular lenguaje Pascal. Su nombre es un acrónimo de ALGorithmic Oriented Lenguage.

Algoritmo (Algorithm). Conjunto de operaciones de cálculo desarrolladas en una secuencia preestablecida y conducente, en un número finito de pasos, a la resolución de un problema específico (en contraposición a heurístico).

Algorítmico, lenguaje. Lenguaje diseñado para definir algoritmos.

Almacenamiento externo. También llamado memoria externa. Dispositivo periférico del ordenador diseñado para almacenamiento de datos (discos, cintas magnéticas, etcétera). En contraposición a la memoria interna del ordenador (RAM, ROM, etc.).